MODULAR HOME MONITORING SYSTEM

Robert Simon, Robert Short, Gary Leutheuser

UNIVERSITY OF CENTRAL FLORIDA Group 7

Table of Contents

1. Executive Summary

Many homeowners have a need for a way to monitor the status of their dwellings while they are away. A familiar example of a product that fulfills this need is a home security system. Consisting of a base station and a number of linked sensors, these notify the homeowner (and possibly the local police) of any activity that might indicate a break-in. However, there are also other applications. Consider a home that is frequently left unoccupied for long periods of time, such as a vacation home. If, for example, a water pipe were to burst, the owner would need to know about it, preferably before significant damage were to occur. There are many similar scenarios (such as a fire or a power outage, to name a few) which the owner would like to know about without having to resort to constant personal supervision.

If an integrated, modular system for monitoring various sensors and generating notifications based on the gathered data were available, the task of taking care of a property would be greatly simplified. Perhaps more importantly, being able to remotely verify the safety of a structure would do wonders for the owner’s peace of mind. Ideally, such a system would have configurable alerts, and a real-time monitoring function, so that the user could both receive alerts when an abnormal and possibly dangerous condition is encountered (such as a fire) or simply view sensor readouts to ensure that everything is working properly (by checking the temperature to ensure that the air conditioner is functioning, for example).

There are many systems on the market which try to fulfill this need. Unfortunately, most of them fall far short of the ideal. Commercially available security systems will monitor motion detectors or door sensors and alert the police when tripped. However, many do not alert the homeowner (giving him or her an opportunity to interrupt what may be a false alarm). Most are also tied to a particular service provider, and will not function at all unless a significant monthly fee is paid. Almost none have any support for environmental sensors (humidity sensors, smoke detectors, etc.). Dedicated environmental sensor systems exist, but these do not support security-related sensors. At present, the best option for anyone who desires both security and environmental monitoring is to maintain and install a separate system for each. This leads to a significant increase in both complexity and cost.

This is where the Modular Home Monitoring System comes in. This system fills the need of a low cost, easy to use, and modular wireless sensor network for home monitoring. The MHMS provides Carbon Monoxide, smoke, and humidity detection as well as a live video feed of the users home for their peace of mind. Alerts are sent to the user when anomalies are detected and the user can log into the online web interface to view current sensor readings and the video feed. At any time, the user can choose to customize the setup of sensors in their home according to their need, up to 7 devices. This gives the user total control over the coverage of their home. For example, a particular user may want a CO sensor in the garage and bedroom, a smoke detector in the kitchen and living room and a humidity detector in the basement, all in addition to the Camera/Main Control Unit that watches the main living space. This is only possible with the MHMS.

2. Project Description

2.1. Personnel

The project consists of three University of Central Florida seniors:

- Robert Simon, Computer Engineering
- Robert Short, Electrical Engineering
- Gary Leutheuser, Electrical Engineering

2.2. Project Specifications

The following subsections describe our project requirement specifications, both hardware and software. These will be the culmination of research into constraints and how they affect our project.

2.2.1. Hardware Specifications

2.2.1.1. Weight & Construction

The following hardware weight and construction specifications were selected after a careful review of the system's needs and research into similar projects.

- The Main Control Unit shall weigh no more than 2 pounds.
- The Main Control Unit shall occupy a volume of no more than 200 cubic inches, with dimension limits of 4 inches by 4 inches by 2 inches.
- The sensor pods shall weigh no more than 1 pounds each.
- The sensor pods shall occupy a volume of no more than 24 cubic inches, with dimension limits of 4 inches by 3 inches by 2 inches.

2.2.1.2. Functionality & Performance

The following hardware functionality and performance specifications were selected after a careful review of the system's needs and research into similar projects.

- The system shall be able to sense levels of carbon monoxide, smoke, and humidity.
- The hardware shall be able to support home monitoring via a camera.
- Sensor pods shall send their measurements or alerts to the Main Control Unit wirelessly, and the Main Control Unit shall then send data to the User Interface wirelessly over the Internet.
- The sensor pods shall be able to detect when a "Hazardous Situation" occurs. A hazardous situation is when Carbon Monoxide is detected above 100ppm, Smoke is detected at all, and when humidity is detected above 70%.

2.2.1.2.1. Main Control Unit

The following main control unit hardware specifications were selected after a careful review of the system's needs and research into similar projects.

- The Main Control Unit (MCU) shall be able to communicate to the sensor pods as well as the Internet wirelessly.
- The MCU shall collect measurement data from sensor pods and send them to the user interface.
- The MCU shall have at least a 400MHz clock rate, 250MB of RAM and be capable of running a full operating system.
- Under typical use, the CPU on the MCU shall not run at more than 85% usage.
- Under use, the CPU temperature on the MCU shall not exceed 80c.

2.2.1.2.2. Sensors

The following sensor hardware specifications were selected after a careful review of the system's needs and research into similar projects.

- The sensors shall provide a common interface for communication with the Main Control Unit.
- The Sensors shall be constructed of a common board called the "Interface Board" and another board unique to the different sensor pods.
- Each sensor pod shall be dedicated to measuring and communicating one quantity (CO, Smoke or Humidity).
- The humidity sensor shall provide a minimum resolution of 1% relative humidity (RH).
- The humidity sensor shall have an absolute accuracy of ±5% RH.
- The CO sensor shall be able to detect at least 50ppm of CO.

2.2.1.2.3. Camera

The following camera hardware specifications were selected after a careful review of the system's needs and research into similar projects.

- The image sensor shall provide at least 3 Mega Pixels of image quality.
- The image sensor shall be able to provide an image stream of 320x240 images at 1 fps.
- The Camera shall be integrated onto the Main Control Unit and not require additional power or control circuitry.

2.2.1.3. Energy

The following energy specifications were selected a after of careful review of the system's needs and research into similar projects.

- The Main Control Unit shall have no more than a 20W power consumption.
- The Interface Board shall be powered by battery or 5V power supply from a standard US electric outlet and provide 3.3V and 5V power to components and add on boards.
- The MCU shall be powered by standard 120 V, 60 Hz power, typical of American homes.

2.2.1.4. Economic & Time

The following Economic and Time specifications were selected after of careful review of national standards, the system's needs and research into similar projects.

- The cost of the entire system including all the sensors, main control unit, camera, software and licenses shall be less than $500.
- A prototype of the MHMS shall be able to be constructed in less than 3 months.
- Quality management shall be enforced so if the MHMS goes into production, economic benefits can be obtained. (ISO 10014:2006)
- Cables in the MHMS, whether for programming microcontrollers or connecting components, shall be economically optimized (BS IEC 60287-3 2:2012).

2.2.1.5. Ethical, Health, and Safety

The following Ethical, Health and Safety specifications were selected after of careful review of national standards and research into similar projects.

- The sensor pods and main control unit shall adhere to the FCC RF safety policies.
- All radio frequency devices used shall comply with Part 15 of Title 47 of the Electronic Code of Federal Regulations.
- ESD safety guidelines shall be followed in the production and handling of the MHMS's semiconductor components. (IEC/TR 62258-3 Ed 2.0 b:2010)
- Any Batteries used in the MHMS's sensor pods shall adhere to standard safety regulations and pose no threat to safety or health to the user or their home. (IEC 60086-1 Ed. 11.0 b:2011)
- Rechargeable batteries used by the MHMS's sensor pods shall be recharged via a charging circuit capable of preventing overcharging. (IEC 60086-4 Ed. 4.0 b:2011)

2.2.1.6. Environmental, Social, and Political

The following Environmental, Social and Political specifications were selected after of careful review of national standards and research into similar projects.

- The MHMS shall pose no threat to the natural environment and care shall be taken to ensure prototype construction and disposal practices are environmentally friendly. (RC 14001:2013)

- The MHMS shall be socially responsible, this includes data privacy protection and overall consumer protection. (ISO 26000, ISO 10001, ISO 10002, ISO 10003, ISO/TS 10004)

2.2.1.7. Manufacturability and Sustainability

The following Manufacturability and Sustainability specifications were selected after of careful review of national standards and research into similar projects.

- Custom Printed Circuit Boards that are a part of the MHMS shall be designed for manufacturability as per standard PCB manufacturer guidelines.
- Passive and active Surface Mount Devices shall be constrained to standard imperial physical package sizes.
- All electrical components shall be sourced from major online vendors such as Digikey and Mouser to ensure short lead times and availability of components.

2.2.2. Software Specifications

2.2.2.1. Functionality

The following Software Functionality specifications were selected after of careful review of the system's needs and research into similar projects.

- The cloud software shall provide user access via web user interface to view sensor health readings and video camera data at all times.
- The cloud software shall alert users via text message when a hazardous situation is detected.
- The cloud software shall include a data store that saves sensor data for up to 30 days for viewing by user.
- All communication and sensor information on the sensor pod shall be processed by the sensor's microcontroller before being transmitted to the main control unit.
- Sensor pod shall send information on its battery life to be displayed on the web user interface.
- Once a sensor pod is paired to the Main Control Unit, it shall automatically connect to the network on power up and begin functioning.
- The sensor pod to Main Control Unit communication shall be reliable.
- Sensor pod to Main Control Unit transmissions shall follow all the requirements of the standard chosen.
- The Main Control Unit shall translate sensor pod data to IP/TCP packets for routing to the cloud software.
- The Main Control Unit shall connect to the internet via a wireless protocol.
- The main control unit shall use a scanning approach to read sensor data outputs

2.2.2.2. Performance

The following Software Performance specifications were selected after of careful review of the system's needs and research into similar projects.

- All off system software will run in a dedicated cloud platform and the platform shall have at least a 99.99% uptime.
- There shall be no more than 10 seconds between when the system cloud software receives a hazardous situation alert and a text message is sent to the user.
- The web user interface shall display the video feed with no greater than 10 seconds of latency.
- The sensor pods shall operate with optimal power consumption via software sleep modes and short packet transmission.
- The sensor pods shall run size and power optimized code on their microcontroller.
- Sensor pods shall communicate to the Main Control Unit with at least 98% reliability.
- A network latency of no greater than 5 seconds shall exist between any of the sensor pods and the main control unit.
- The Main Control Unit shall be ready to send and receive transmissions within 20 seconds of power on.
- The Main Control Unit shall translate sensor pod data to an internet ready packet within 2 second.
- The Main Control Unit shall have wireless capabilities to transmit data at 1Mbps or greater.

3. Research

3.1. Architecture

There are a few different overall architectures being used for Wireless Sensor Networks. The architectures differ on how the sensor modules send their data to a backend service. The two big types of architectures are ones with low powered sensors and a nearby more powerful control unit, and ones with high power independent sensors capable of connecting to the internet directly. Both of these architectures are very different in terms of network topology and computational distribution. We discuss the differences in the following sections.

3.1.1. Network Topology

Network topology for the low power sensors with control unit architecture resembles the typical star topology with the sensors connected to the control unity with a low powered wireless communication protocol like Bluetooth or ZigBee.

The control unit is generally connected to a wall outlet power source so it can use whatever it wants for communication. One project used the GPRS network for secure communications independent of if the location had an internet location or not. [1] The benefits to this design are inexpensive/low power sensors, simple communication, and flexibility with the control unit choice and design. Some disadvantages are the need for additional hardware (the control unit) and limited range between the sensors and the control unit.

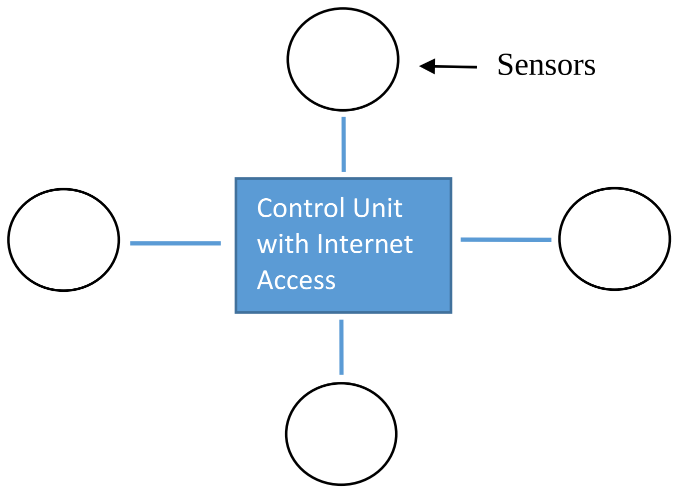


Figure 1 - Star Topology Example

The second type of architecture is very “Internet-of-Things” style and involves having independent internet enabled sensor systems and no control unit.

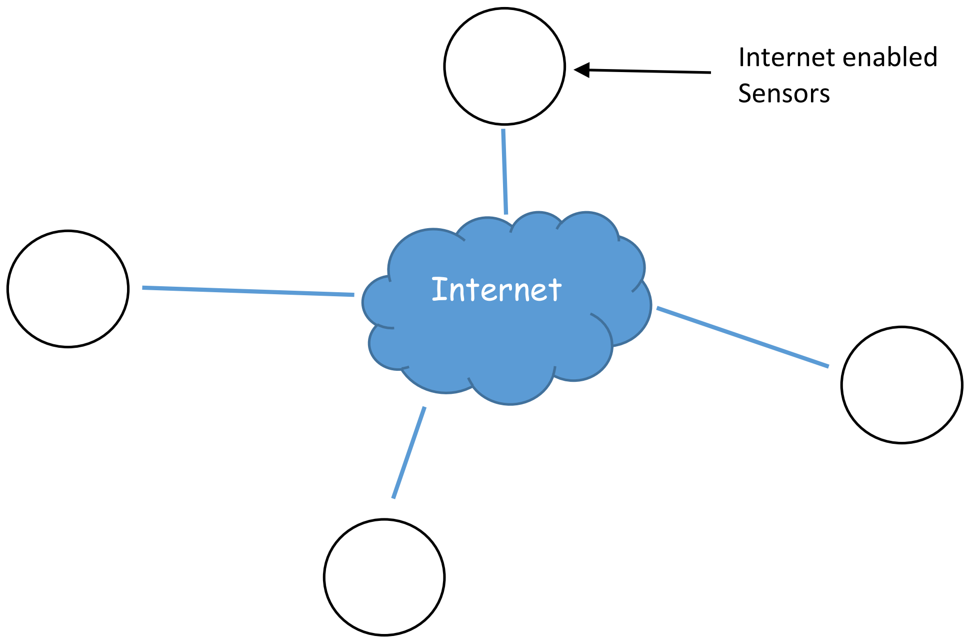


Figure 2 - Example Internet of Things Topology

This type of architecture has some advantages including large range, low hardware cost, and easy setup/configuration. Some disadvantages include a higher level of software complexity, much higher power usage on sensors, and potentially overwhelming user’s home network. This would be an excellent architecture to use for a very large house with a decent WiFi system already in place.

3.1.2. Computational Distribution

Computational load distribution is something that is very important to understand for low powered embedded systems. The question that comes up is where do you do the bulk of the processing? You can do it on the embedded microcontrollers, on a central controller or on an online server. According to research presented at IEEE’s INFOCOM 2005, it is necessary to have a balance between computation on a sensor node and the amount of time the node spends transmitting. Considering we have a battery powered system with a microcontroller and some form of wireless RF transceiver, we know that the RF transceiver is going to spend much more power during its operation than the microcontroller will in running commands. So for optimal battery use, we want to use the on-system microcontroller to monitor when it is necessary to send data and to get the sensor’s data into as few bits as possible for transmission. The less bits in a message to be sent the less RF energy will be needed and thus we have a longer lasting battery. [2]

So the closer to the sensor that data processing gets done, the better battery performance we will see. However, any other type of heavier processing should be taken to either the main controller of the network or onto an internet server. The controller that is embedded within the wireless sensor network should be relatively powerful but should not be relied upon as a processing powerhouse. The main controller can be used for short term statistical analysis and tasks like that but processing time should be reserved for receiving data from sensor nodes and transmitting data to user via internet. Long term statistical analysis and data crunching should always be done on a server or cloud infrastructure. [3]

Looking back to the Star Architecture or Internet of Things Architecture, we can see the differences in how data is computed is different. In the Star architecture, it is much like the above examples where sensor data is processed directly on the embedded microcontroller and any other computation is moved to the central controller and then to the internet server for increased computational capability. In the Internet of Things Architecture, the central controller is removed completely so there has to be a lot more processing on the nodes for the data to be useful for the software running on the online server.

3.1.3. Sensor Interface

There are several types of sensor interfaces explored in research today. Some popular ones include fully embedded, the BASE Platform, and partially modular interfaces. The sensor interface is how the sensor and the microcontroller communicate. A fully embedded interface has everything on one board and involves having a generic and simple analog and digital interface to a socket where you can have different types of sensors. This type of interface is best used for simple sensors and different sensors will require different software loads on the microcontroller. This approach is the most simple to implement because the hardware will all be the same but will have no decoupling between any of the components. The BASE platform system has two boards for one sensor package. Each board has its own microcontroller and are connected via a bridge. One board has all the communication and data storage while the other has the sensor and any conditioning circuitry. This allows for maximum uncoupling of communication protocol, firmware and sensor technology however it does increase the cost per unit by quite a bit. Lastly we have the partially modular interface which involves two boards like the previous system, except the main board is the one with the CPU and all the communication hardware, and the secondary board only contains the sensor and its conditioning circuitry. The different sensors have their own conditioning circuitry and thus can each have their own unique boards, but they can all work with the same main board with the microcontroller and communication protocol. This allows for some decoupling between all the sensors and the controller firmware and for more complex sensors to be used. [4]

3.2. Main Control Unit (MCU)

The following sections describe findings from industry research relating to the picking and usage of a Main Control Unit for the MHMS to communicate with the sensor pods.

3.2.1. Communication Protocols

A communication protocol is a system of rules that two entities follow so that information can be transmitted between them. There are so many communication protocols being used in the field that naming them all would take pages, but overall, all protocols are defined in a layered fashion and provide all or part of the services specified by the OSI Model.

The OSI Reference Model characterizes and standardizes communication functions without having to care about the underlying structure and technology. Each layer has its own functionality and in general each layer exists to serve the one above it. The physical layer is the lowest layer and deals with the transmission and reception of the raw bit stream over a physical medium like a wire or the air. The Data Link Layer provides error free transmission of data frames from one node to another over the physical layer. The Network Layer controls which physical path data should go based on network conditions, basically it does the lower level routing functions. The Transport layer involves higher level routing functions, it is the first layer that has to know the big picture of routing a packet from one node to another. The transport layer has to ensure messages are delivered without error and in sequence. There are technically two other layers above the transport layer called the session and presentation layer but many people exclude them in their discussion of the OSI model. The Application layer is the highest layer and serves as the intermediary between the user and the network functions. The application layer would include your web browser, Outlook, or file transfer program. [5]

The benefit of choosing a protocol based on the OSI model is that you get interoperability between independent systems regardless of what technologies are used in both. This would be very useful in a wireless sensor network application where you may have multiple sensors using different transmission technologies. One packet may contain a low battery warning and that message can be understood by all nodes if they all use the same protocol.

Some basic elements of a protocol are the data formats, address formats, address mapping, routing, and error detection, acknowledgement of reception, sequence control and flow control. A useful communication protocol must define as many of these elements as possible. [6]

3.2.1.1. Sensor-to-MCU Communication

The following sections describe the possible Protocols and Data Formatting one could use for a Wireless Sensor Network as found through industry research.

3.2.1.1.1. Protocols

Many competing protocols exist for the purpose of wireless communication. The primary differences among them are: power usage, range, data rate, resistance to interference, and of course, simplicity of implementation. The ideal protocol would maximize all of these properties, but in reality trade-offs are inevitable. Each of the protocols that we considered for our application prioritizes these qualities differently, and must be evaluated against the needs of our project in order to select the best candidate.

3.2.1.1.1.1. Bluetooth

Bluetooth (IEEE 802.15.1) is a wireless communication standard that operates on the 2.4 GHz ISM band. This standard was originally designed to be a wireless replacement for RS-232 cables and has come a long way since its inception in 1994. [7] Bluetooth is a packet-based protocol meaning data is transmitted between packets in discrete data units called "packets". Bluetooth is generally used in applications were two devices need to connect with minimal configuration. It is because of this that we are very partial to using Bluetooth for our Sensor-to-MCU communication. The most common Bluetooth network architecture is a master-slave configuration where we have one master and up to 7 slaves. It is possible to have more than 7 slaves but complexity of the network increases as you have to include more masters or have some of the slave nodes be "masters" for some period of time. This would put a 7 sensor limit on our project unless we implement a "master-hopping" algorithm.

Several revisions of the Bluetooth protocol have brought about great increases in range, data transfer speeds, and power utilization. Theoretically, with V3.0 + HS, Bluetooth can get a maximum data rate of 24 Mbps. However, the latest Bluetooth version, V4.0, added Bluetooth Smart/ BLE (Bluetooth Low Energy), which is the low energy protocol stack that can be utilized with Bluetooth devices. BLE lowers range and data throughput for a massive savings in power consumption. Even so, BLE offers 0.27 Mbps data transfer rate and a 50 meter range. [8] A study by Microsoft Research found that a popular Bluetooth module (TI CC2450) found that when comparing BLE, ZigBee, and ANT standards, BLE had the most efficient power usage with a sleep current of 0.78 uA and awake current of 4.5 mA using a 3.3V supply. [9]

Every Bluetooth device has a unique, 48-bit address which are used to set up connections. To set up a connection, first a device must be in "Discoverable mode" where it transmits the device name, device class, list of services and some technical information. A device can then pair with the discoverable device by creating a link key. Two devices are paired if they store the same link key and can then communicate with one another. Because Bluetooth operates in the ISM band between 2.402-2.480 GHz so to avoid interference, the band is divided into 79 channels of 1MHz each. Using Frequency-hopping spread spectrum technology, a device changes channels around 1600 times per second to achieve sufficient reliability. [10]

3.2.1.1.1.2. WiFi

802.11, otherwise known as WiFi, is a family of specifications for implementing WLAN communication in several frequency bands. The most popular protocols in the WiFi family for embedded applications are 802.11g and 802.11b. [11] Both 802.11g and b operate on the 2.4 GHz ISM band and use direct-sequence spread spectrum and orthogonal frequency-division multiplexing signaling to mitigate interference potential from other devices operating within the ISM band. 802.11g has the advantage in theoretical data rate with a threshold of 54Mbps, while 802.11b has a theoretical data rate of only 11Mbps. In our application we do not necessarily need a high speed data rate since we are only going to transmit small packets of data very scarcely. Channels in the 2.4GHz band for WiFi are split into 11 22MHz wide channels in the United States. However, because of signal attenuation, stations can only use every fourth or fifth channel without overlap.

802.11 microcontrollers tend to have higher power consumption when compared to other wireless modules. TI has a popular WiFi enabled microcontroller called the CC3200. According to a current measurement test by TI, the CC3200 has a TX current of 281mA, an idle current of 0.66mA, and a RX current of 62.2 mA. [12] This is with the chip being capable of a TCP throughput of 13 Mbps. As we see here, with a higher current draw comes a much higher data rate than some of the other wireless protocols.

3.2.1.1.1.3. Z-Wave

Z-Wave is a lesser known wireless specification that runs in the sub-gigahertz frequency band. It was designed with reliability, low latency and low power consumption in mind. Z-Wave is advertised to have a data rate of 100kbit/s which is the lowest of all the wireless specifications explored in this section. One of the more interesting aspects of Z-Wave is that it does not compete with Wi-Fi, Bluetooth or the other 2.4 GHz band specifications so interference is not as much of an issue. The range of a Z-Wave device is approximately 30M. [13] A typical Z-Wave controller, the ZM5304 has a RX current of 32mA and a TX current of 36mA @ 0dBm, so in terms of power consumption the controller is very good.

Building a Z-Wave network is relatively easy, in fact it is the easiest network to set up among the 5 specifications in this section. All you have to do is “pair” devices which involves pressing a sequence of buttons on the controller and on the device being added. Once a device is paired, it will always be recognized by the controller. A Z-Wave network is identified by a Network ID and a device is identified by a unique Node ID. The Network ID and Node ID comes from the controller and gets stored on the nodes. Two Nodes with different Network ID’s cannot communicate with each other. A source-routed mesh network topology is used which allows devices to route communications, thus effectively expanding the network range far beyond the radio range of a single unit.

3.2.1.1.1.4. ZigBee

ZigBee is a wireless communication specification that is based on an IEEE 802.15.4 standard. IEEE 802.15.4 defines the physical layer and media access control for low data rate wireless networks. ZigBee takes that foundation and adds the network and application layers. ZigBee network topologies are similar to the other wireless specification, a ZigBee Coordinator is the master for a network and can act as a bridge to other networks. A ZigBee router is a device that passes data between devices and can likewise run applications. Lastly a ZigBee End Device has the lowest level of functionality, it should draw just as much power as it needs to communicate to the coordinator or router. ZigBee End Devices thus have a very long battery life, with the ZigBee foundation requiring a certified device have at least a two year battery powered lifespan. [14]

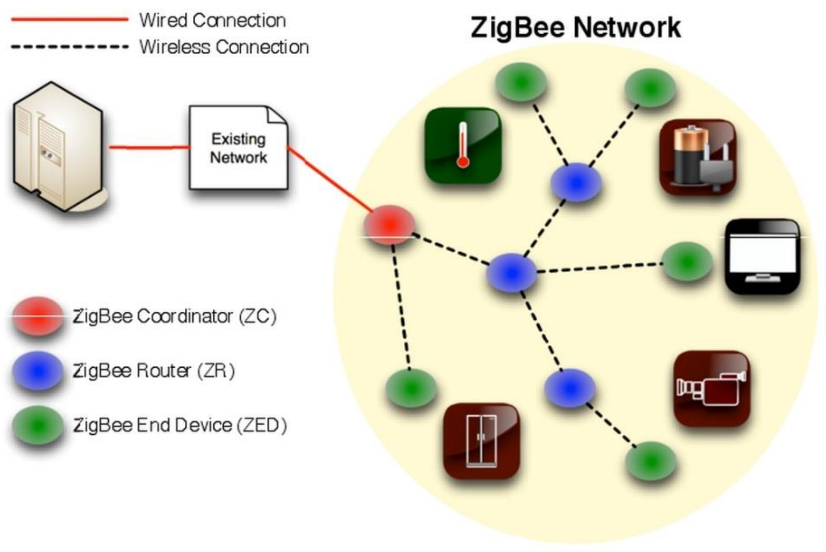


Figure 3 - Example ZigBee Network, reprinted with permission from M. Collotta, G. Pau, V.M Salerno and G. Scata and intechopen.com

ZigBee was designed for low rate, low latency applications like sensors and was a major competitor to Bluetooth in the early 2000’s. ZigBee has since fought out Bluetooth for the edge in LAN network use while Bluetooth remains the most popular for PANs. One smart home device developer, called GreenPeak released an opinion describing ZigBee as complementary to Bluetooth. They stated that while ZigBee is a LAN networking technology having the ability to cover an entire house with an unlimited amount of nodes, Bluetooth is a PAN technology with the ability to only cover a few meters and with severe limitations on the amount of nodes. All of this while having fundamentally the same cost. [15]

A popular ZigBee SOC from TI, the CC2531 is specified to have an Active RX current of 24mA @ -79 dBm, an Active TX current of 29mA @ 1dBm, and a PM1 (lowest power mode) current draw of 0.2mA with a 4us wake time. Physical signal range is limited to 10-20 meters (actual) and data rate is defined to be 250 kbit/s. [16] With these features it is no wonder so many have flocked to ZigBee for sensor network and smart home applications.

3.2.1.1.1.5. GPRS

General Packet Radio Service (GPRS) is the packet based data service used by 2G and 3G cellular systems. It was designed to allow cell phones to transmit IP packets wirelessly to networks like the internet on the GSM network. GPRS realistically provides 15-40 kbps data rate and is a best-effort service, meaning there is no guaranteed Quality of Service. Unlike standard GSM which used circuit switching technology, GPRS used packet switching which allowed for better integration with the rest of the internet. GPRS has a channel bandwidth of 200 kHz, can operate on several frequency bands, and uses Gaussian Minimum Shift Keying (GMSK) modulation. Throughput and latency depend on the number of users using the service which is unique when compared to the other 4 wireless specifications reviewed in this section. GPRS is quite unique in that a device can connect to the GSM network (and thus the internet) without needing to have additional hardware. [17]

One of the most popular GPRS modules is the SIM900 by SIMCom Wireless Solutions Co. According to their data sheets, one module can operate on a 3.2-4.8V supply and has a 1 mA sleep mode current draw. To send data the module uses between 500 and 2000 mA of current. [18] GPRS sends data in “bursts” which are 0.577 mS long. A single burst carries two blocks of information which totals to 144 bits per burst. For our application, the high current draw and potentially high latency would be a major concern in terms of battery life and user notification speeds.

3.2.1.1.2. Data Format

A message data format adopted for use within a wireless sensor network must be as simple as possible to reduce network traffic and conserve battery, while still being as informative as possible. One data format specifically designed for Wireless Sensor Networks is called the Sensor Information Data Format (SIDF) from the Tampere University of Technology. Addressing nodes in a sensor network under the SIDF involves three values, the network, node, and sensor numbers organized in a three level addressing approach “network.node.sensor”. Each ID in each level must be unique to make unambiguous addressing possible. In SIDF there are three message types, the measurement message, request message and response message. A measurement message must contain the network.node.sensor address as well as information on what is being measured and the value of the measurement. A request message happens when the client wants to begin receiving real time measurements from a node and contains the address and requested measurement. Lastly we have a response message which returns certain values depending on if the request was successfully received or not.

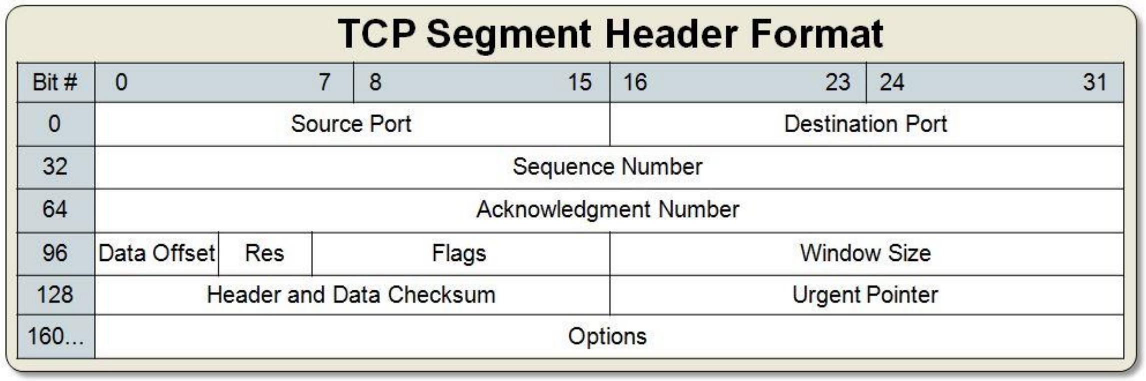
SIDF sends data messages in XML format for simple parsing however the XML format does have considerable overhead. If an application has a message size constraint due to power or number of nodes then it may make sense to actually build your own format. Taking the IP datagram as an example, we see that one datagram is split into two parts, the header and the payload. If a custom data format is desired for a specific application the developer would have to decide what fields go in the header and payload to maximize the information density of a message.

3.2.1.2. Main Control Unit-to-User Communication, TCP/IP vs UDP

TCP/IP are the two most important protocols in the Internet Protocol Suite. This is the set of protocols used by many communication systems, but most importantly, it is the one adopted by the Internet. TCP/IP defines how data should be organized, packetized, addressed, transmitted, routed and received across a network. The key to the Internet communication is abstraction and encapsulation. The network is abstracted into different layers that encapsulate one another following the OSI model. Data comes from an application, goes down to subsequent layers, gets transmitted over a physical medium and then goes up the layers in another device until the data reaches its destination. This abstraction makes it possible for data to be transmitted between two systems with different technology.

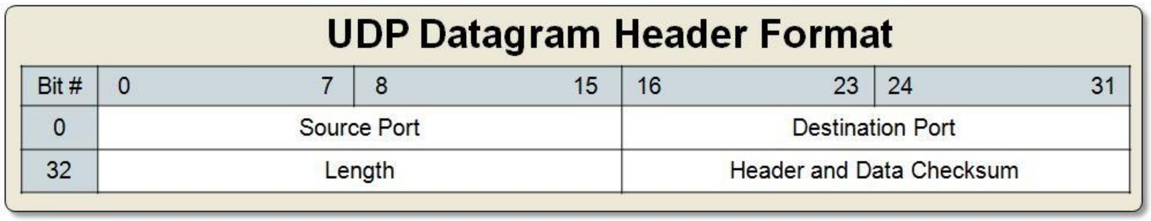
Transmission Control Protocol (TCP) is part of the transport layer and is responsible for breaking data into small packets, as well as assembling the packets when they arrive. Internet Protocol (IP) which is part of the network layer is responsible for the communication between computers. It addresses data packets and is the one responsible for sending and receiving data packets over the network. There are many other protocols that work with TCP/IP like HTTP (web browsing), FTP (file transfer) and SMTP (email). [20] A TCP/IP packet includes headers for both the TCP packet and IP datagram. Because of encapsulation, the entire IP datagram will be in the data field of the TCP packet. The IP datagram header includes data like the source and destination IP address, the protocol used, and the time to live. A TCP Packet header contains the source and destination ports, sequence number, acknowledgement number, the size of the TCP buffer on the host, and the TCP checksum. [21]

TCP/IP provides a connection oriented, reliable and ordered system of sending a stream of data over a network, however there are times where we do not need much reliability and could do without the overhead that TCP/IP comes with. The Universal Datagram Protocol (UDP) is a protocol that provides connectionless datagram communication with an emphasis on reducing latency at the cost of reliability. Just by looking at the UDP header we can see the differences compared to TCP/IP. A UDP packet header only contains the source/destination port, the length of the message and the message checksum. The UDP header is only 8 bytes compared with at least 20 bytes in a TCP header. UDP has no handshaking dialogs so any unreliability in the underlying network is transparent to the user application. However UDP is popular in applications were errors occur infrequently or if error checking is done at the application layer. The massive reduction in network latency makes UDP popular for time sensitive applications such as streaming video or voice. On top of these, UDP is very easy to work with since the protocol is so simple one can learn to work with UDP packets much quicker than TCP. [22]



TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset Res	Flags	Window Size	
128	Header and Data Checksum		Urgent Pointer	
160…	Options			



UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

Figure 4 - TCP vs UDP Header

3.2.2. MCU Options

When it comes to picking out a main control unit for a wireless sensor network, one has to consider the suite of features provided by the hardware. Since building a system powerful enough to run the type of software needed to run the network would be very difficult, most people end up buying either a Raspberry Pi, or a BeagleBone Black. Both of these are microprocessor boards meaning they are essentially small, fully featured computers capable of running full operating systems.

The BeagleBone Black comes with a 1GHZ ARM processor, 512 MB of RAM, 4GB of flash storage, a USB port, Ethernet, HDMI and 2x 46 pin headers. The BeagleBone has the leg up when it comes to I/O capabilities, with 7 12-bit ADC, 8 PWMs, and 4 UARTs. This impressive I/O setup means the BeagleBone will be much better for driving motors and building small connected electronic projects with lots of interfaces. [23]

The newest edition of the Raspberry Pi is the Raspberry Pi 2 Model B. This model comes with a 900 MHz quad core ARM processor, 1 GB of Ram, 4 USB ports, 40 GPIO pins, HDMI, Ethernet and a dedicated camera CSI[2] connector. [24] Interfaces include 1 UART, 1 SPI Bus and 1 I2C Bus. The lack of hardware interfaces makes the Raspberry Pi 2 not a great option for building large interconnected projects right out of the box, but the other hardware specs make the Raspberry Pi a more powerful processing unit. One downside of the Raspberry Pi unit is that the hardware is not open source, meaning you cannot pull up the schematics or PCB layouts if you every wanted to incorporate additional functionality. BeagleBone is completely open source so you can view all the details of the chip and even build your own if you had the resources.

3.3. Sensors

The aim of our project is to consolidate and present environmental information that is of relevance to homeowners in a convenient form. Accordingly, a robust and varied sensor network is key to accomplishing our goals. We have chosen a small set of sensors that can gather useful information when deployed in a typical home.

3.3.1. Carbon Monoxide Sensing Technologies

There are many potential sources of Carbon Monoxide (CO) gas in the typical home. Possible sources include: fuel-burning appliances (gas stoves, furnaces, grills, etc.), generators, and automobiles. When properly used and maintained, fuel-burning appliances will not introduce harmful amounts of CO into the home, but in reality these appliances are often improperly maintained, and can become a danger.

CO gas harms humans by binding to hemoglobin to produce carboxyhemoglobin (COHb), which is unable to carry oxygen. The symptoms induced by continuous exposure to various concentrations of ambient CO are summarized below.

CO Concentration (ppm)	COHb percentage	Symptoms
35	< 10%	Headache and dizziness within 6 to 8 hours
100	> 10%	Headache in 2 to 3 hours
200	20%	Headache in 2 to 3 hours; loss of judgement
400	25%	Frontal headache in 1 to 2 hours
800	30%	Dizziness, nausea, and convulsions within 45 minutes; insensible within 2 hours
1600	40%	Headache, tachycardia (rapid heart rate), dizziness, and nausea within 20 minutes; death in less than 2 hours
3200	50%	Headache, dizziness, and nausea in 5 to 10 minutes; death within 30 minutes

6400	60%	Headache and dizziness in 1 to 2 minutes; convulsions, respiratory arrest, and death in less than 20 minutes
12800	> 70%	Death in less than 3 min

Source: "Carbon monoxide poisoning". Journal of Emergency Nursing: JEN: Official Publication of the Emergency Department Nurses Association

There were 146 unintentional non-fire CO poisoning cases in the United States, of which 79 percent occurred in the home. For this reason, the U.S. Consumer Product Safety Commission (CPSC) recommends that at least one certified (by Underwriters Laboratories, International Approval Services, or the Canadian Standards Association) CO detector be installed in the hall outside of each sleeping area in a residence [25].

3.3.1.1. Opto-Chemical

Opto-chemical detectors are some of the simplest detectors available. They consist of a pad impregnated with a chemical that changes color upon exposure to CO gas. Detection can be automated by using an LED and photodiode to measure the reflectance. These detectors only indicate the presence or absence of CO; they cannot measure concentration. Furthermore, they must be replaced periodically (approximately every four months). For these reasons, this type of detector has largely been supplanted by the newer types discussed below.

3.3.1.2. Electrochemical Cell

Electrochemical CO detectors consist of two electrodes immersed in an electrolyte, which is exposed to ambient gases. CO is converted to carbon dioxide at one electrode, and the current through the cell is directly proportional to the CO consumed.

3.3.1.3. Semiconductor

Semiconductor CO sensors consist of a layer of tin dioxide on a ceramic substrate. Absorbed CO reduces the resistance of the semiconductor. The sensor must be heated significantly above room temperature in order to function, and the power requirements of the heater must be taken into account. Accordingly, semiconductor CO sensors are typically powered from the mains.

3.3.1.4. Biomimetic

Biomimetic CO detectors consist of a gel matrix which contains a mixture of compounds designed to mimic the action of hemoglobin. The matrix exhibits a change in absorbance proportional to the CO concentration, which can be measured with an LED/photodiode combination. The advantage of these detectors is that they are highly resistant to false positives. Ethylene has been shown to cause false positives in electrochemical and semiconductor type detectors, but does not trigger biomimetic ones [26]. However, ethylene is rarely encountered in significant quantities in a residential setting, and such would likely never arise in practice.

3.3.2. Smoke Detecting Technologies

According to the National Fire Protection Association (NFPA), properly installed and maintained smoke detectors reduce the risk of dying in a house fire by half [27]. While prompt warning of potential fires is valuable, a balance must be struck between sensitivity and false positives, which lead to “nuisance alarms”. Frequent, spurious alarms can force residents to disconnect or otherwise disable the alarm, negating any protective effect. For this reason, any viable smoke sensor must quickly and accurately detect fires (preferably before they have a chance to grow significantly), but also be resistant to false positives.

3.3.2.1. Optical

Optical smoke sensors measure the density of smoke by using an LED/photodiode pair to measure the proportion of light that is scattered by particles in the air. If this scattered light exceeds a set threshold, the sensor triggers. This type of sensor is more sensitive to fires in the smoldering stage, one of the earliest stages of a fire [28].

3.3.2.2. Ionization

Ionization smoke sensors operate by ionizing a sample of air located between two electrodes, and measuring the current through them. Ionization is accomplished by an alpha particle source, usually a sample of Americium-241. When smoke particles enter the detector, they bond to the free ions and reduce the current between the electrodes. In order to compensate for ageing of the alpha source, changes in air pressure, changes in temperature, and other factors that affect ionization rate, a sealed reference chamber is also included. When the current through the primary chamber drops below that of the reference, the sensor is triggered. This type of sensor is most sensitive to fires in the flaming stage [28].

3.3.3. Glass Break Detecting Technologies

Assuming that the doors of a home or business are secured and locked, windows represent some of the weakest points in a structure’s physical security. It is relatively simple for an intruder to break a pane of glass in order to enter a secured area, and the sound of breaking glass is often not loud enough to attract attention from bystanders, assuming any are present. Ideally, a more durable material could fulfill a similar role to glass (admitting light and allowing vision) without its fragility. Unfortunately, such materials are not available at a reasonable price for consumers. Glass break sensors represents a compromise solution, which detect when a monitored window has been shattered and generate an alert.

3.3.3.1. Physical

Physical glass break sensors attach directly to a window in order to detect shattering. These sensors most commonly take the form of fragile conductive elements adhered to the glass surface, such that any breach will cause an open circuit, and trigger the sensor. Physical sensors are very simple and reliable, but cannot be easily reset (the conductive elements must be repaired if damaged) and cannot cover an area greater than one pane of glass; multiple sensors must be installed in order to cover an entire building.

3.3.3.1.1. Window Foil

Before the development of acoustic sensors, the most common method of securing windows from forced entry was “window foil”, consisting of a thin metal foil adhered to a window. In the event that the window was shattered, the foil would break (breaking the sense current) and trip the alarm. For most applications, a strip of foil around the edge of each window provided a good compromise between security and aesthetics. However, it was possible for a skilled intruder to cut a hole in the center of the window and short out the foil, allowing the glass to be broken with impunity. To combat this, the foil would have to repeatedly traverse the entire pane, or a thin wire mesh (operating on the same principle as the foil) would be used instead.

An undeniable advantage of window foil is its simplicity; an alarm circuit need only monitor for a substantial increase in resistance. This system is also more localized, and can determine which and how many monitored windows have been breached. This also makes it resistant to false positives, since it will not react to other nearby glass shattering (as an acoustic sensor might), only panes to which it is applied.

There are also distinct disadvantages to this type of protection. The chief difficulty is that it is difficult to maintain. A careless window cleaner can damage or destroy the applied foil, rendering it useless. A related issue, but still important problem is that window foil requires significant labor to apply, exacerbating the problem of repairing an accidentally damaged sensor.

3.3.3.2. Acoustic

In contrast to the physical sensors discussed earlier, acoustic sensors are triggered by the sound of glass breaking, rather than physical force. The obvious advantage of this type of sensor is that they are easier to install, and require no modifications to the windows they protect. The corresponding disadvantage is their lower specificity; they only determine that some glass has been broken, not where the breach has occurred.

3.3.3.2.1. Band-limited microphone

The cheaper and simpler sensors in this category use a band-limited microphone, tuned to a frequency range produced by shattering glass but uncommon otherwise. They have the advantage of simplicity, but are more prone to false positives than sensors which conduct a more detailed analysis of the received audio.

3.3.3.2.2. Spectral analysis

The more complicated sensors are based around spectral analysis of the received audio. They calculate the spectrum and compare it to a reference, generating a similarity score which is then compared to a threshold value, set by the designer. If the threshold is exceeded, the sensor trips. Some may include heuristics to reduce the incidence of false positives; testing will be required to determine if this is necessary.

3.3.4. Humidity Sensing Technologies

3.3.4.1. Capacitive

The capacitive sensor is just that: a capacitor whose dielectric constant (and thus capacitance) changes value depending on the humidity of the air that it is in. Capacitive humidity sensors are limited in that they must be relatively close to their signal conditioning circuitry, due to the parasitic capacitance contained in the physical connection to the sensor. The technical characteristics of this technology are summarized below in Table 1 - Capacitive Humidity Sensor Characteristics [29].

Table 1 - Capacitive Humidity Sensor Characteristics†

Characteristic	**Value**
Transfer function	Near-linear
Humidity Range	0 % to 100 % RH
Accuracy	± 1.7 % to ± 5 %
Sensitivity per % RH	0.31 pF to 4 pF
Response time	< 5 s to 50 s
Supply Voltage	2.3 V to 15 V
Cost	$2 to $80

†Data obtained from DigiKey inventory

3.3.4.2. Resistive

The resistive sensor contains a resistive element that varies inversely with humidity. Life expectancy for a resistive sensor in the home is much greater than 5 years [29]. In addition, large temperature fluctuations can cause the sensor to shift, although this should not normally be an issue in an air-conditioned home [29].

Table 2 –Resistive Humidity Sensor Characteristics†

Characteristic	**Value**
Transfer function	Inverse
Humidity Range	5 % to 100% RH
Accuracy	± 3 % to ± 7 %
Sensitivity per % RH	N/A
Response time	~ 60 s
Supply Voltage	100 mV to 24 V
Cost	$4 to $300

†Data obtained from DigiKey inventory

3.3.5. Camera

The addition of a camera system to the suite of sensors adds quite a bit of value: it interacts with one of our strongest and most vivid senses, vision, contributing to a greater feeling of security and peace of mind. It does not do this in an entirely superfluous manner. The saying “a picture is worth a thousand words” is no coincidence. There are countless qualities and characteristics of a scene that would take complicated sensors to process and transmit the information contained – instead, just as cloud computing offloads processing tasks to more capable warehouse computing systems, our system also offloads processing of the inherent qualities of a scene to another, and in many ways more capable, processing system – the human brain.

3.3.5.1. Complete Video Systems

There are many complete video systems featuring a variety of interfaces. One popular example is the “webcam” – a small, high quality video camera with a common interface such as USB or WiFi.

There are also many existing video solutions designed specifically for security purposes. Many of these cameras come in large quantities, to be distributed around a property for static monitoring, or they feature a gimbaling system, allowing a smaller number of cameras the ability to pan and tilt in order to continuously “scan” the scene for disturbances.

The typical characteristics and features of the range of systems include:

- 720p or 1080p resolution
- Auto focus
- Roughly 2 in. x 2 in. x 1 in. in size (webcam) to 5 in. x 5 in. x 5 in. (security camera)

- Face tracking, if applicable
- USB or WiFi interface
- Requires interfacing with a PC running a supported OS
- Ability to move in 2 axes, if applicable
- IR LEDs for low-light conditions
- \$20 to \$250

3.3.5.2. Image Sensor Technologies

3.3.5.2.1. CCD

The charge-coupled device (CCD) image sensor consists of an array of CCDs: MOS capacitors which share a continuous insulator [30]. This shared insulator, along with the close proximity of the gates, allows transfer of charge between "cells" – these cells can be configured to permit light to introduce the electron-hole pairs, and thus detect the amount of light striking a physical section of the sensor.

This technology has a number of notable performance benefits [30]:

- Low "dark current" (power consumption when the sensor is not exposed to light)
- Simple extraction of cell data (charge may be coupled out to single nodes regularly spaced instead of requiring individual cell addressing)
- Stable, compact, and robust, by design

Some of the characteristics of the CCD image sensor are summarized in Table 3.

Table 3 – CCD Image Sensor Characteristics†

Characteristic	**Value**
Pixel Size	7.4 µm x 7.4 µm to 10 µm x 10 µm
Sensor Pixel Count	19,200 to 1,046,360
Frame Rate	20 to 90
Supply Voltage	3.3 V to 22 V
Cost	\$10 to \$5,500

†Data obtained from DigiKey inventory

3.3.5.2.2. CMOS

The CMOS image sensor has become predominant in the recent market, leading to lower costs and more advantage over the CCD image sensor [30]. The CMOS image sensor is built much like an electronic memory, and consists of an array of PN junctions, which generate current when light is incident on them [30]. The “cells” are accessed line-by-line, unlike the CCD sensor. This trades off size for speed.

At present, due to economy-of-scale principles and the popular nature of CMOS, it can offer a large number of advantages over CCD sensors [30]:

- Lower cost
- Lower noise
- Faster

But still contains some drawbacks [30]:

- Larger pixel size
- Lower sensitivity
- Lower dynamic range

The characteristics of the CMOS image sensor are summarized in Table 4.

Table 4 – CMOS Image Sensor Characteristics†

Characteristic	**Value**
Pixel Size	1.4 µm x 1.4 µm to 25 µm x 25 µm
Sensor Pixel Count	12,348 to 26,214,440
Frame Rate	3 to 580
Supply Voltage	1.8 V to 5.5 V
Cost	$0.93 to $16,000

†Data obtained from DigiKey inventory

From a cursory review of this single supplier’s inventory, the CMOS image sensor is vastly superior to the CCD image sensor for our purposes, if only because it is readily available, with a wide selection at reasonable costs for low-volume production.

This cost also poses potential for a standalone image sensor to be less costly than an existing camera system. The low upfront cost for the sensor is, however, quickly mitigated by the cost of packaging, interfacing, and integrating required (something that is avoided with an existing system).

3.3.5.3. Image Processing

While the image processing capabilities of our brains appear to far exceed that which can be accomplished in electronic designs, a human has a severe limitation: we would much prefer not reviewing hours of footage in order to perform identification of potential disturbances in a scene. In any continuous monitoring system, a need for supporting processing systems has developed and work has been done on bringing electronic systems into the realm of image processing so disturbances can be found easily, bringing specific, small periods of time into consideration for more capable human analysis.

3.3.6. Analog to Digital Conversion

An Analog to Digital Converter (ADC) is an electronic device that converts a continuous physical electrical quantity to a digital value. The physical quantity is generally voltage however it can also be current. An analog signal is sampled at a specified "sampling rate" to get the digital value of the amplitude of the signal. The higher the sampling rate is, the more data points we have per second and thus the more accurate representation we have of the analog signal. However one issue engineers run into tis that the higher the sampling rate, the more space it takes to store the data. One way to get a good balance between storage and quality is the Nyquist Theorem. The Nyquist theorem states the sampling rate on an ADC must be at least two times the highest frequency you want to capture. For example, if we were recording music, the highest frequency humans can hear is 20 kHz so if we wanted to record music with good storage to quality balance we would need at least at least 40 kHz sampling rate. [31]

The resolution of an ADC is another important consideration. Most embedded microcontrollers have either a 10 or 12 bit resolution on their ADC's. Let's consider a 10 bit ADC, that means it can capture up to 2^{10} (which is 1024) discrete digital values. In reality the value of the reading would be between 0 and 1023 which corresponds to the amplitude of the input analog signal. The higher the resolution, the more accurate your ADC will be as well. If we had a microcontroller with a maximum analog input of 3V (which is typical) we can see the effect of resolution on our accuracy with some math.

$$10\ Bit \rightarrow 1023\ steps \quad \rightarrow \quad \frac{3V}{1024 steps} = 2.93mV\ per\ step$$

$$12\ Bit \rightarrow 4095\ steps \quad \rightarrow \quad \frac{3V}{4095 steps} = 732.4uV\ per\ step$$

As we can see, there is a significant accuracy difference that comes with just two more bits of resolution. [32]

3.3.7. Power

For low-power sensor modules, there may be an advantage to utilizing battery power, even if the sensor is fixed in place and theoretically could be powered through a wall socket or hard-wired to the building's mains power. The motivation for such a design would usually be to avoid monopolizing a wall socket (or otherwise requiring the use of an unsightly power strip) or to simply avoid the bulk and hassle of an external power supply.

A typical example of this usage would a be a Hall effect sensor, commonly used in commercial security systems to detect if a door or window has been opened. In such a situation, a primary (that is, non-rechargable) battery would be preferred, since infrequent maintenance is the main goal, and the higher cost of batteries of the device's lifetime is outweighed by the inconvenience of manually recharging it. A battery that is typically used for door and window sensors is the CR123A, a 3 V lithium primary battery.

Another important use for battery power is as a backup, even in cases where mains power is easily available. In security applications, a temporary power interruption should not interrupt monitoring. For this reason, a reserve battery is necessary for the base station at a minimum, and should be present in all of the security sensors (motion sensors, camera, etc.) as well.

Rechargeable technologies are shown in Table 5 - Rechargeable Technologies.

Table 5 - Rechargeable Technologies

Chemistry	**Voltage**	**Energy Density**	**Notes**
Sealed Lead Acid (SLA)	1.5 V	60 Wh/kg, 100 Wh/L	While SLA batteries have a poor energy density, they are considerably less expensive than other battery types, and require little maintenance. For stationary, high-power applications, such as an uninterruptable power supply (UPS) unit intended for a server, SLA batteries dominate. The heavy metal content presents an environmental hazard, but recycling largely negates the impact.
Nickel-Cadmium (NiCad)	1.2 V	90 Wh/kg, 210 Wh/L	Despite their low energy density, NiCad batteries are typically more robust than other chemistries. They have a long lifetime (many charge-discharge cycles), and can supply high currents. However, the heavy metal content presents an environmental hazard.
Nickel-Metal Hydride (NiMh)	1.2 V	125 Wh/kg, 400 Wh/L	NiMh batteries are inexpensive (though not cheaper than NiCad or SLA) and relatively light (though heavier than lithium batteries). They are environmentally friendly.
Lithium-Ion (Li-ion)	3.6 V	240 Wh/kg, 550 Wh/L	Li-ion batteries are relatively expensive, but offer high energy density. However, a malfunctioning

			cell can cause a fire, presenting a safety hazard.
Lithium-Ion Polymer (Li-poly)	3.6 V	260 Wh/kg, 540 Wh/L	Li-poly batteries have similar characteristics to Li-ion, but offer a moderate increase in energy density and a moderate increase in price.
Lithium Iron Phosphate (LFP)	3.3 V	108 Wh/kg, 220 Wh/L	LFP batteries have a lower energy density than other lithium batteries, but are safer and have longer lifetimes.

Primary (Non-Rechargeable) Technologies are shown in Table 6 - Primary (Non-Rechargeable) Technologies.

Table 6 - Primary (Non-Rechargeable) Technologies

Chemistry	**Voltage**	**Energy Density**	**Notes**
Alkaline	1.5 V	125 Wh/kg, 400 Wh/L	Alkaline batteries are some of the lowest cost primaries. However, they do not perform well at low temperatures and have a high internal resistance.
Lithium Iron Disulfide	1.5 V	310 Wh/kg, 560 Wh/L	
Lithium Manganese Dioxide	2.9 V	240 Wh/kg, 500 Wh/L	The commonly-used CR123A is an example of this battery chemistry.
Lithium Sulfur Dioxide	2.8 V	265 Wh/kg, 400 Wh/L	
Lithium Thionyl Chloride	3.6 V	520 Wh/kg, 1050 Wh/L	

Another power consideration is that of regulation. To power the various components that interface with the sensors, a multitude of power sources (primarily different voltage levels able to source different amounts of current) are needed.

There are several common types of voltage regulators available, the two most prevalent being linear regulators and switching regulators. These technologies are compared in Table 7.

Table 7 – Voltage Regulation Technologies†

Characteristic	Linear	Switching
Output Voltage	0.5 V – 40 V	0 V – 80 V
Output Current	600 µA – 5 A	1 µA – 10 A
Efficiency	Low	High
Cost	Low	High

†Data obtained from Mouser inventory

The efficiency and cost values require a little extra explanation. The terms are meant to be interpreted as relatively high and low, compared to the other technology. The main cost driver of switching regulators is extra components (including an inductor). Also, the efficiency of switching regulators is highest for high load currents – the load current of a small microprocessor is unlikely to push it very far into its efficient range.

The linear regulator is most efficient when the input voltage is close to the desired output voltage, but in general will be lower than that of a switching regulator. Given the comparable current and voltage outputs, the cost and efficiency dominate the selection of the proper regulator for this project.

The MHMS base station will be powered by a standard USB wall adapter, and has no power requirements beyond those of the Raspberry Pi model B board (maximum current draw of 700 mA at 5 V). However, a battery backup is desirable in order to keep the system operational during power outages. The battery does not need to be as large as the above figures would suggest; testing indicates that 500 mA is a reasonable estimate of the current draw for a moderate CPU load. Taking this into account, we can derive a simple formula for the useful lifetime given by a particular battery capacity, assuming that the battery outputs the necessary 5 V. This formula is only a rough estimation, as it does not take into account the effects that temperature, discharge rate, or other factors have on battery lifetime.

$$Lifetime\ (hours) = \frac{Battery\ Capacity\ (mAh)}{500\ mA}$$

A large number of competitively-price batteries with USB output (accepted by the Raspberry Pi) are available for purchase, due mostly to the large market for portable cell phone rechargers. The standard USB interface will allow the MHMS to operate with any of these. The charge available from these products is typically denoted milliamp-hours, so we can obtain a rough estimate of the lifetime granted by a particular unit, and tailor our selection to the needs of the particular installation.

The Interface Board (IB) which hosts the RFduino and supplies power to the sensor boards, needs to contain a regulated 3.3 V supply for both internal and external use. A provisional power budget for a battery-powered IB is in Table 8 - Battery-Powered IB Power Budget. The maximum power allotted for battery-powered sensor boards is much lower than mains-powered, to ensure acceptable operating time between recharges.

Table 8 - Battery-Powered IB Power Budget

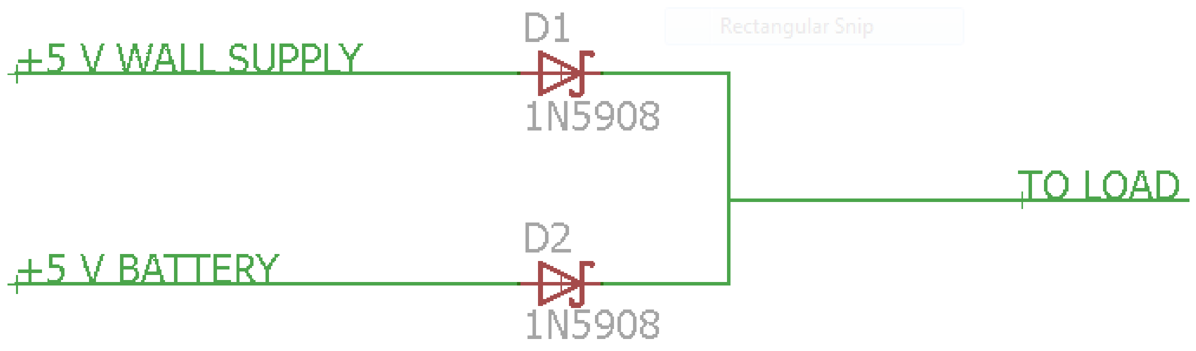
Component	**Max Current Draw (mA) - 5.0 V**	**Max Current Draw (mA) - 3.3 V**	**Max Component Power Usage (mW)**
RFduino		16	52.8
Sensor board		20	66
		Max Total Power Usage (mW)	118.8

A power budget for a mains-powered IB is also provided in Table 9 - Mains-Powered IB Power Budget.

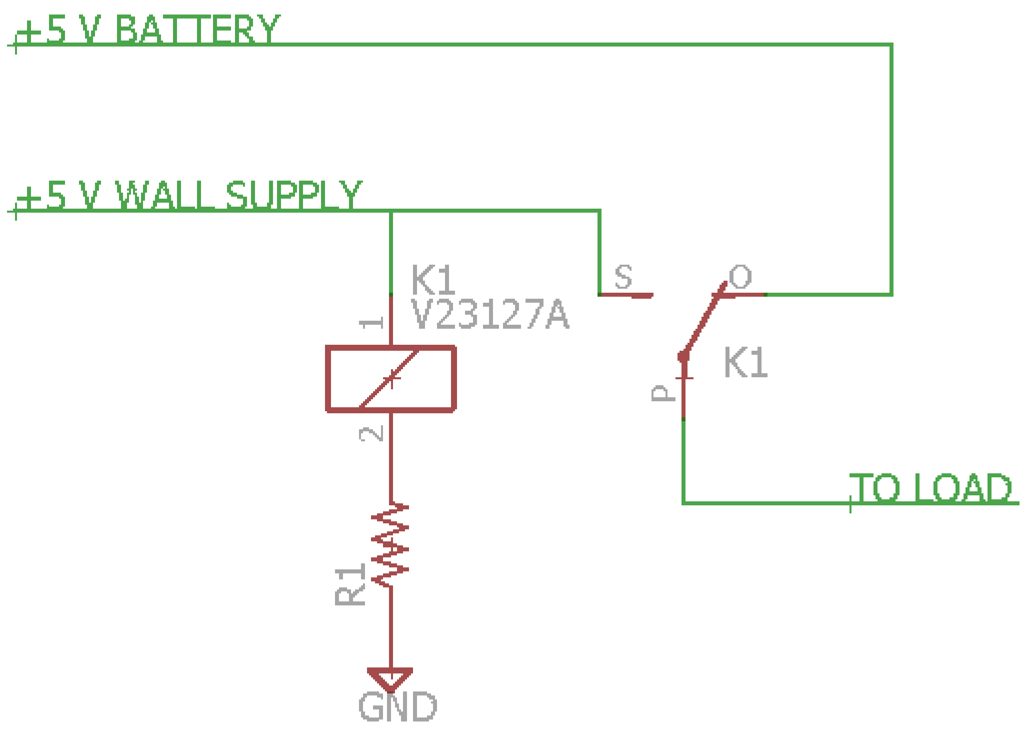
Table 9 - Mains-Powered IB Power Budget

Component	**Max Current Draw (mA) - 5.0 V**	**Max Current Draw (mA) - 3.3 V**	**Max Component Power Usage (mW)**
Rfduino		16	52.8
Sensor board	200		1000
		Max Total Power Usage (mW)	1052.8

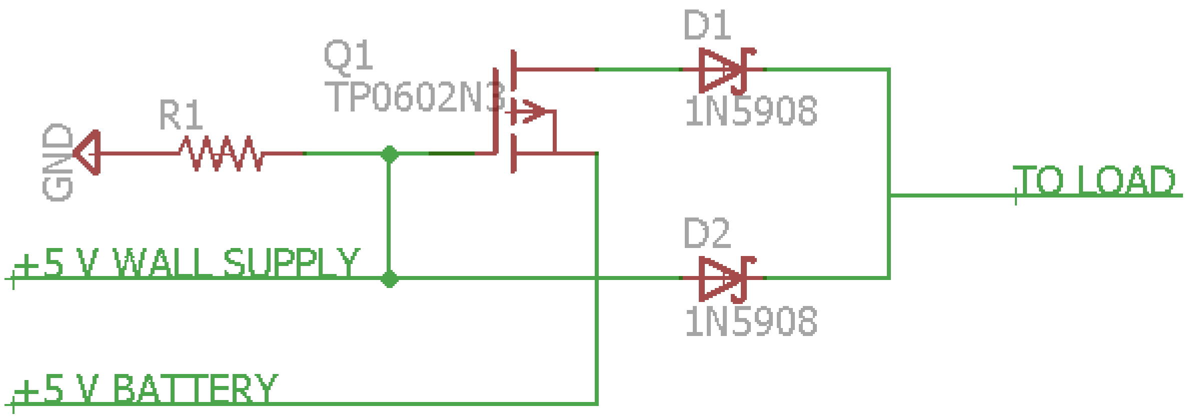
Although the interface board hosting some sensor types (specifically, the carbon monoxide detector) will always be powered from an outlet under normal operating conditions, it is desirable that the sensors also have capability to run from a battery. Aside from allowing the system to function in a power outage, this will simplify the design by only requiring one IB design, rather than separate boards for battery-powered and mains-powered devices. In order to function in the event of a power outage, an automatic battery failover circuit is needed to switch the batteries into the supply circuit when wall power is not present. The simplest and cheapest means of achieving this goal is by a diode-OR circuit, illustrated below.



This circuit is one of the simplest possible implementations, and cheapest in terms of both component cost and board space required. These two metrics are particularly important, since one interface board will be needed for each sensor, and it is desirable to minimize module size and total cost. However, the diode configuration does not respect the priority of the power sources; in particular, if the wall supply voltage drops below the battery voltage (as might occur when the supply is under heavy load), then some current will be drawn from the battery. This behavior both drains the reserve supply unnecessarily, and imposes additional charge-discharge cycles which can reduce battery lifetime. One possible workaround is to adjust the wall supply voltage to be above the battery voltage under expected load conditions.



Another low component count solution is to connect a SPDT relay as shown above, with the supply from the battery connected to the normally closed side of the relay, and the wall supply connected to the normally open side. When the wall supply is present, it will energize the coil, and switch the battery out of the load circuit. If the wall supply fails, the relay will again close and the battery will be able to supply the load after a brief interruption. Resistor R1 is necessary to limit the current through the coil. This method requires few parts, but relays tend to take up a large board area.



A slightly more complicated configuration is shown above. The gate of the p-channel MOSFET (which controls current from the battery) is high when the primary supply is active, but is pulled low by the resistor when the primary supply turns off, allowing the battery to supply power. Diode D1 is necessary to prevent uncontrolled charging of the battery through the MOSFET's body diode, and diode D2 prevents the battery from pulling the gate high. This solution solves the problem associated with the diode-OR configuration, at the cost of a higher component count (which increases overall cost and board size). However, it still uses less board area than the electromechanical option.

For our Interface Board, we would like to use the last circuit shown above. While a diode-OR circuit may function in practice, it provides little control over battery usage, and depends heavily on power supply characteristics. The relay circuit would be reliable, but large, and may suffer from contact bounce. The p-channel MOSFET switch combines the positive characteristics of the previous circuits, and will be optimal for our design. However, if the board size or component count of this solution prove to be prohibitive, we may fall back to the diode-OR design.

3.4. User Interface

A graphical user interface should be user-centered and be focused on accomplishing user goals as efficiently as possible. Often times, engineers are too concerned on software functionality that they do not give any thought into the design of a user interface, a simple "looks good to me" is all the thought put into an interface. An effective GUI can go a long way in making a piece of software great. But how does one create a good GUI? Research has been happening since the early days of computers into how to make an effective interface and some researches have come up with some principles.

One important thing to remember about user interfaces, especially on the web is that it is the interface specifies the dialog, or style of interaction, between the user and underlying machines. Many styles can be used such as command language, menus, forms, question/answer, direct manipulation, natural language, and windowing systems. [33]

Figure 5 shows some more details on how users interact with their computer's hardware/software.

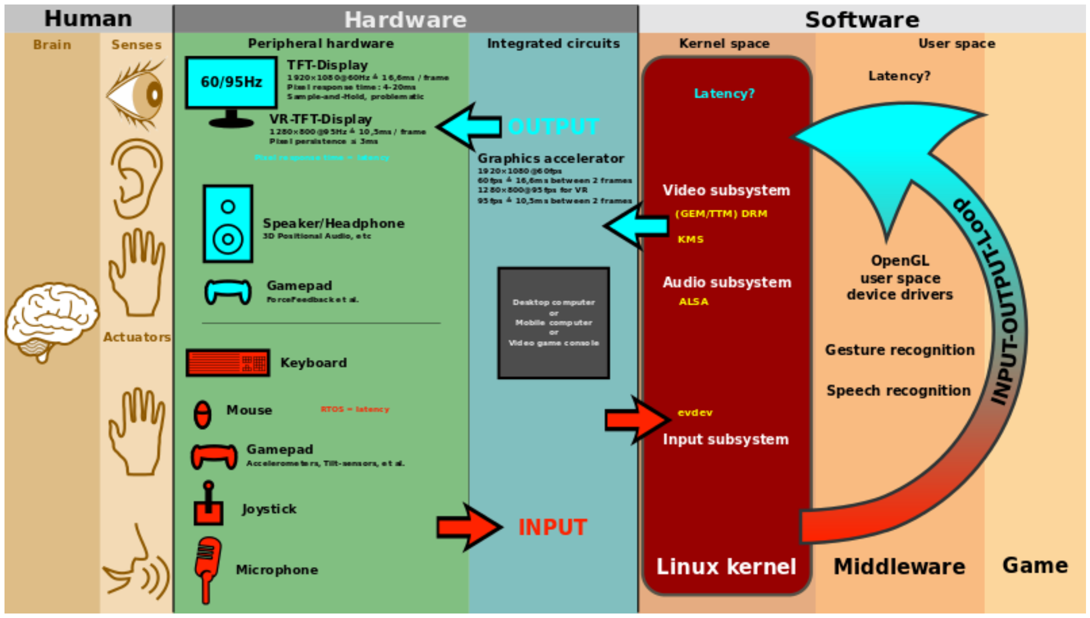


Figure 5 - User Interface Details Used with permission from ScotXW

There are differences is web design and GUI design in general that one must consider. For example, GUI's are built for a specific platform so the designer knows exactly how the GUI will be displayed with all its elements. Web interfaces on the other hand, can be viewed from so many devices, we have smart phones, tablets, smart TV's, desktops, and gaming consoles. Each of these is going to display the web page in a different way so the best way to tackle this problem is to let the presentation of web pages be determined by an interplay of page specifications and the preferences of the client device. [34] Another big difference between GUI and Web interface design is that while GUI's are more of an enclosed experience, a web site that a user visits is more of a "part" of the whole web. It is because of this that an interface made on the web has to follow the general conventions of other sites on the internet. Like how we are used to having the logo of a website be the link to return us to the homepage. Users would get very confused and upset if clicking the logo took you to a completely unrelated page.

There are tons of platforms which cater to web hosting and offer tools that help developers integrate hardware and software systems. One such platform is IBM Bluemix. Bluemix is a cloud platform as a service (PaaS) provider based on Cloud Foundry. Bluemix supports Java, Node .js, Go, PHP, Python, Ruby, and other languages. Bluemix also provides first party services such as Cloudant NoSQL DB, Internet of Things manager, and various Watson services all in addition to regular web hosting. Bluemix would be a great platform to host a user interface for a wireless sensor network because it already supports Internet of Things applications with a native service. Much of the backend work is already done by IBM and you can go straight into building your application and what you want the Node's data to do. The free tier of the Internet of Things service includes 20 active devices, 100 MB of data traffic and 1 GB of data storage. This would be more than enough for developing and prototyping our Home Monitoring System. [35]

A modern web interface is designed with several technologies. The “front end” or, what the user sees, is typically composed of HTML and Javascript code, styled with CSS. HTML is a language that describes web pages using human readable text. Javascript is an object oriented computer programming language typically used to create interactive effects within web browsers. CSS is a language that specifies how documents are presented to users. A CSS document will typically have all the details on how to display the HTML elements on a web page such as color, positioning, format, etc. In addition to the front end, we have what is known as the back end of a web site. Back ends typically use MySQL for database manipulation and PHP for server side scripting. MySQL is the most popular language for adding, accessing and managing content in a database. This content can be pictures, user info, sensor data, really anything. [36] PHP is a server side scripting language that is widely used on the web. PHP code can be mixed with HTML or be used in combination with various engines and frameworks. PHP code can generate a web pages’ HTML, an image, or some other data. The important thing to remember with PHP is that it runs only on the server, and whatever output it generates is sent to the client. [37]

While web interfaces require that users go to the site to see information, sometimes it is necessary to push information to the user. There are generally two ways this is done, through push notifications or through text messages. Push notifications are supported by most smart phones and tablets and are part of a installed app. When the app receives a push notification from a server, it displays the content of the message on the users screen, through a pop up or generally in the notification drawer. Users have a more active roll in setting up these notifications as they have to both install and configure an app to receive these notifications. This may be undesirable if the information is urgent. The other way services get information to users is through the old fashioned text message. There are web services that offer API’s for sending text messages to a user’s phone, one such service is Twilio. Twilio has a host of services including sending SMS, MMS, and voice messages. To use Twilio, you have to request a phone number to be attached to your account. Once it is attached you are given code that can be inserted into your program to initialize the connection to Twilio’s servers. Once the connection is established you can use the Twillio REST API to send a message by making a HTTP POST to Twilio’s servers with the destination phone number and message content. For high priority information, it would serve to reason that text messages are a better suited means of communication because of SMS’s worldwide adoption and its coverage will be much greater than a data network that Push notifications run on.

4. Hardware Design and Schematics

4.1. System Architecture

The Modular Home Monitoring System uses use a Beacon architecture where all sensor pods are broadcasting packets following the IBeacon protocol to the Main Control Unit. The Main Control Unit is then the point where data can be pushed onto the internet and to the Cloud Application. The Sensor Pods wirelessly broadcast to our Main Control Unit (Raspberry Pi 2) via the Bluetooth Low Energy protocol. Functionality is built into the system to allow an arbitrary number of Sensor Pods to run at the same time, however, our prototype will only include one of each of the three types of pods. The sensor pods send constant status messages and monitor the environment when powered on, sending sensor readouts to the MCU. The MCU then sends that data over the internet through WiFi (802.11) to the cloud application that then checks if the sensor data readout meets any danger thresholds. If necessary, the application sends the user a text alert and displays the alert on the web user interface. The camera is connected to the Raspberry Pi via the dedicated camera ribbon cable using the Camera Serial Interface (CSI). The MCU starts streaming image data from the camera when the user logs into the user interface and starts the stream. More details on the design of the sensors and MCU configuration will be discussed in the following sections.

In addition, the sensor pod will consist of a common "interface board" as well as the "sensor board", which houses the sensor itself and all supporting circuitry. This architecture allows for rapid addition of new sensor types and for hardware reusability. This level of detail is depicted in Figure 6 - MCU - Sensor Interface Block Diagram.

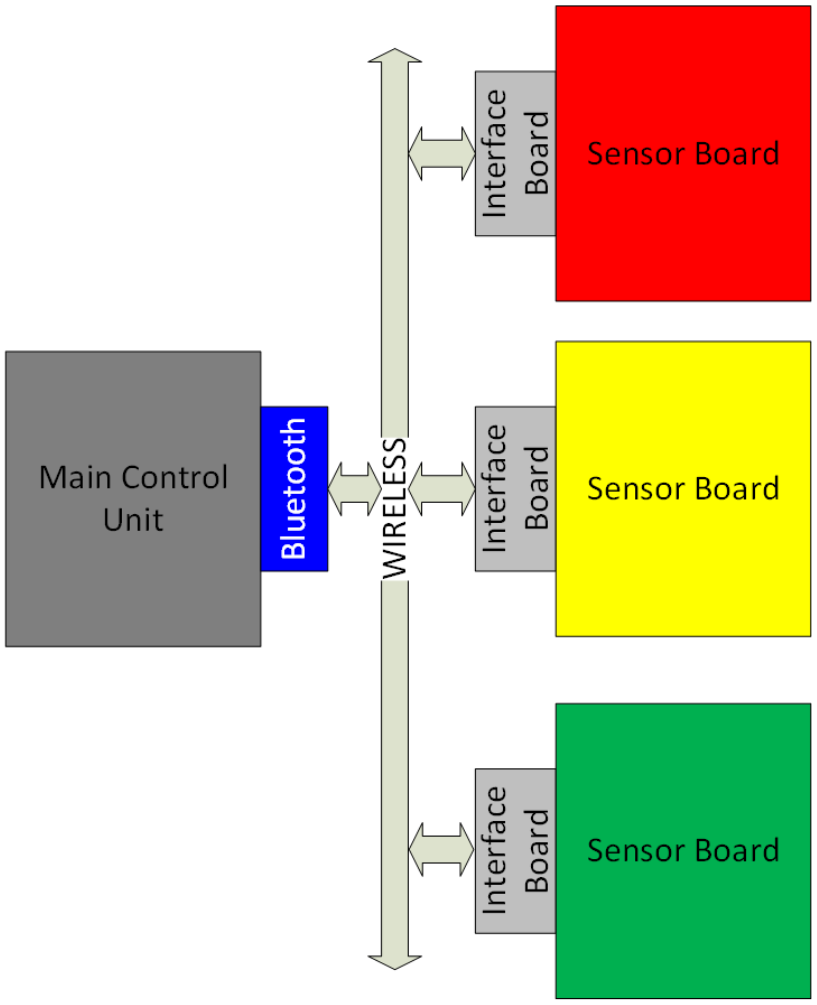


Figure 6 - MCU - Sensor Interface Block Diagram

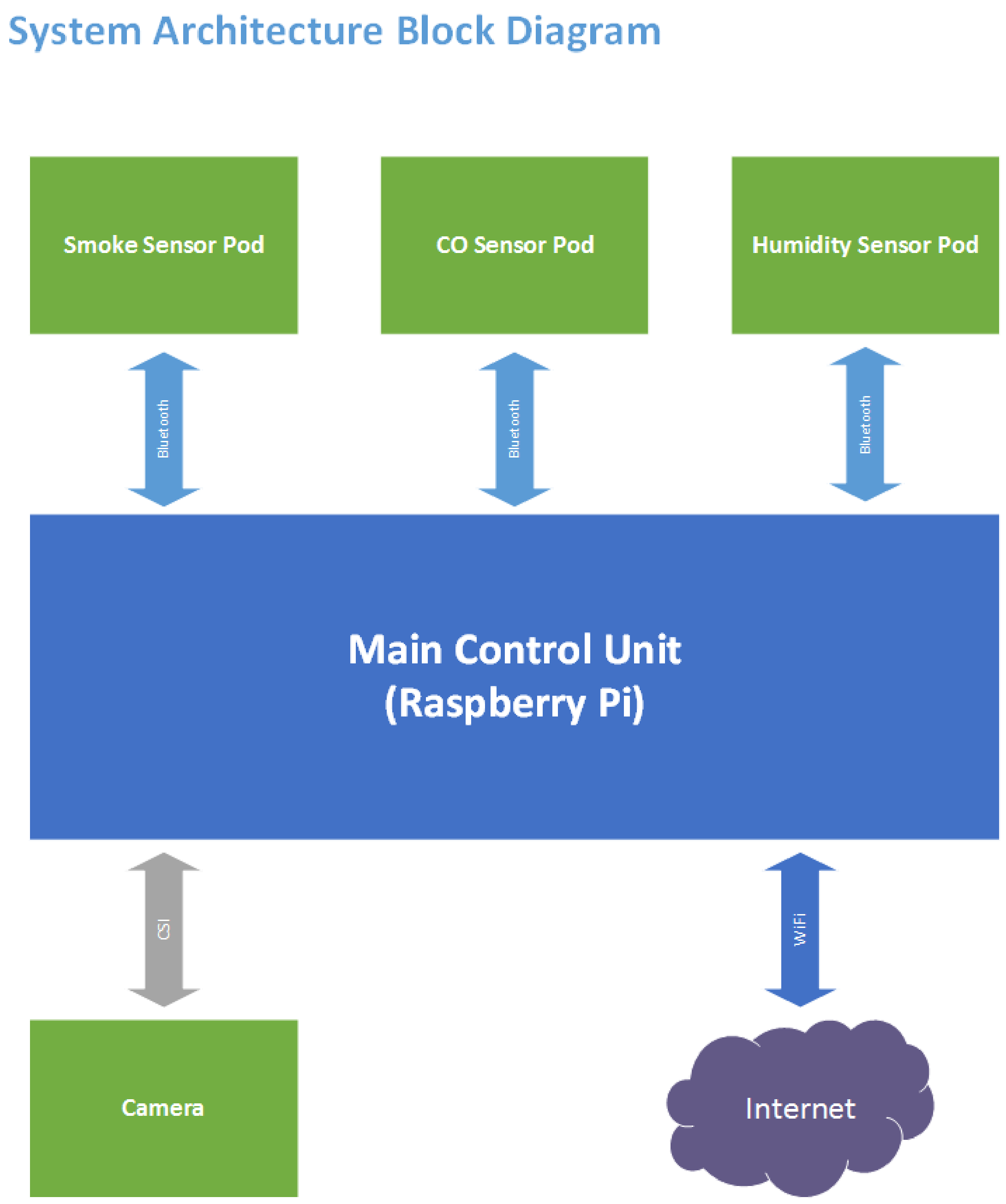


Figure 7 - System Architecture Block Diagram

4.2. Main Control Unit (MCU)

The Main Control Unit for the MHMS is the ever so popular Raspberry Pi 2 model B Micro Processor running the Raspbian operating system. We chose this device above competitors like the Beagle Bone because of its extra USB ports, dedicated camera interface, and its low price point.

This device serves as the “middle man” between the sensor pods and the system cloud software. Data received from the Sensor Pods will not be modified or interpreted, but directed to the cloud software which will then interpret the data and react accordingly. The MCU has several peripherals attached to it, those peripherals being a USB WiFi dongle, a USB Bluetooth dongle, and the 5MP Raspberry Pi camera. The WiFi dongle connects the raspberry pi to the internet through a configured WiFi connection. The Bluetooth dongle acts as the Bluetooth host for the sensor pods. It transmits and receives all Bluetooth data in the system. Lastly, the camera is used by software running on the Raspberry to send an image stream to the cloud software to be displayed on the web user interface.

The Raspberry Pi 2 B has several features that make it a very capable microprocessor system. These features include a 900MHz Quad Core ARM Cortex-A7 CPU, 1 GB of Ram, Micro SD card slot for storage, 4 USB ports and a dedicated Graphics Core. The fact that the Raspberry Pi runs Linux distributions is also a major advantage, especially for development. Operating systems provide a hefty layer of abstraction from that hardware that basic microcontrollers cannot achieve. In addition to abstraction, there is a massive community of users, engineers and tinkerers for the Pi and Linux in general so helpful hands will be in no short supply.

Power is no concern for the MCU as the Raspberry Pi must be powered with a Micro USB 5 V wall power supply at 2 A minimum (for our usage). The drawback to just using a wall plug for the MCU is that if the power in the home goes down, so does all functionality of the Modular Home Monitoring System. However, in the future a battery backup can be added to the MCU to give it a failsafe and several hours of use even when the power goes off.

Main Control Unit Block Diagram

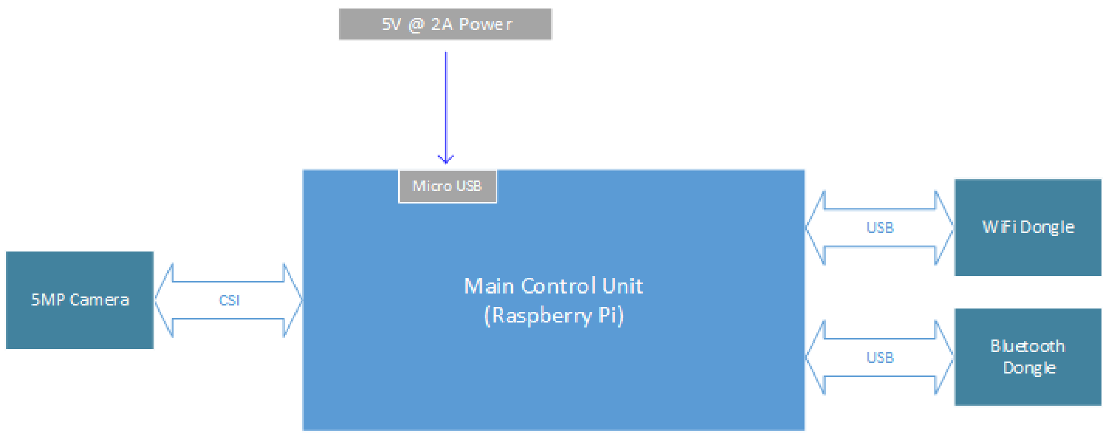


Figure 8 - MCU Block Diagram

4.3. Sensors

The sensors and their interfaces consist of two separate design efforts: that of the interface board, which will be common to all sensors, and that of the sensor board itself, which will feature specific components that vary from sensor to sensor (for example, a sensor with a very low voltage output might require an amplifier to be used. The sensor-specific board would possess this amplifier).

The interface board handles those tasks which are common to all sensors: sending information wirelessly back the Main Control Unit, as well as conversion from analog voltages to digital representations of those voltages, in order for them to be analyzed by the processor and transmitted.

This results in a total number of N + 1 hardware design efforts, where N is the number of sensors (with the exception of the camera, which requires no special interface and does not even pass through an interface board). The + 1 term comes from the interface board itself, which, as mentioned before, is a shared effort. The interface board design details can be found in section 04.3.1 Interface Board, and the specific sensor boards can be found in their respective sections.

4.3.1. Interface Board

The interface board serves the purpose of creating a modular platform between each distinct sensor and the main hub. It accomplished this by keeping its interface simple, and not including sensor-specific hardware in its design. The core of the interface board is the RFduino wireless communication IC, with a built-in antenna. This allows the establishment of wireless data transfer and control of the specific sensors. Because this device is programmable, and contains analog-to-digital converters, the device can be easily reprogrammed to interface with a variety of sensors without requiring a hardware change.

A high-level block diagram of the interface board is shown in Figure 9 - Interface Board Block Diagram.

The next “modular” aspect of the interface board is its generic I/O header. This allows the external sensor to be connected in whatever way is most convenient for that particular sensor. This header will also pass power to the sensor.

In this way, firmware/software changes alone can be created for each distinct sensor, and loaded onto the same shared functionality interface board as necessary, saving hardware costs.

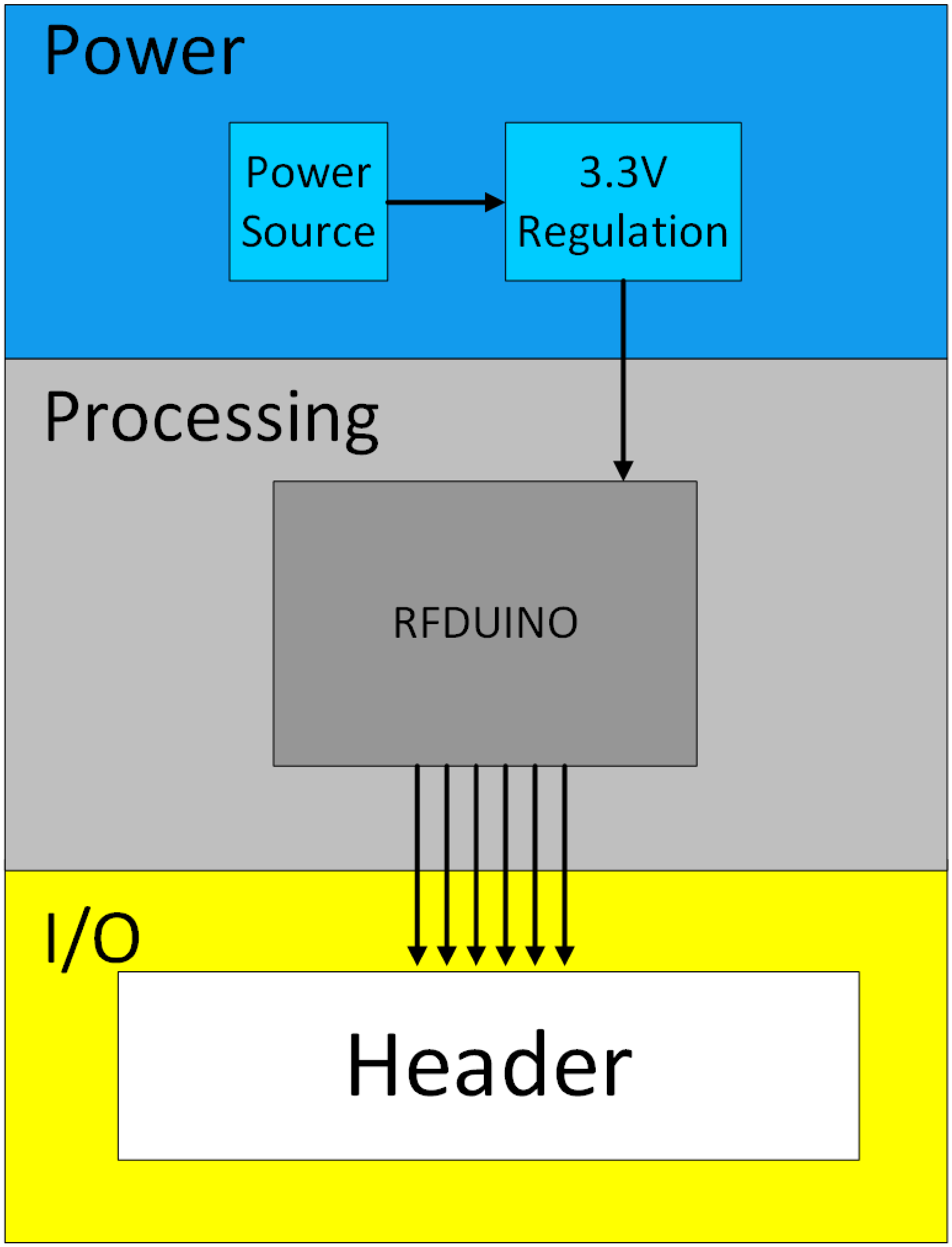


Figure 9 - Interface Board Block Diagram

The schematic for the interface board is shown in

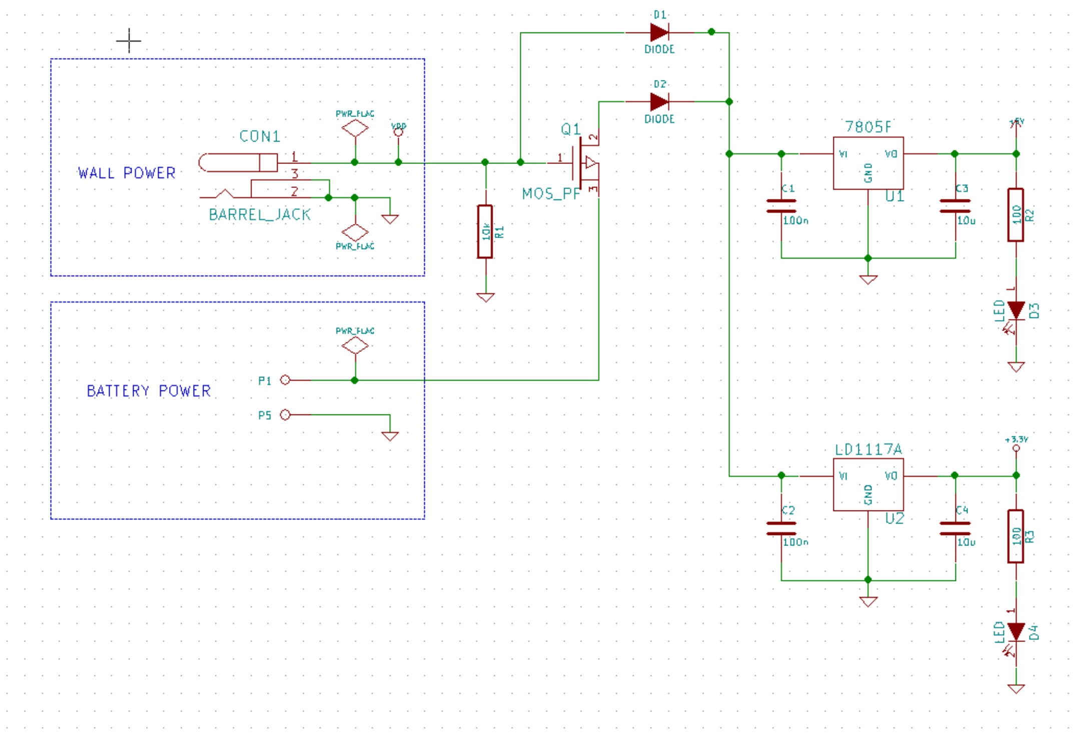


Figure 10 - Interface Board Power and Figure 11 - Interface Board RFduino/Headers.

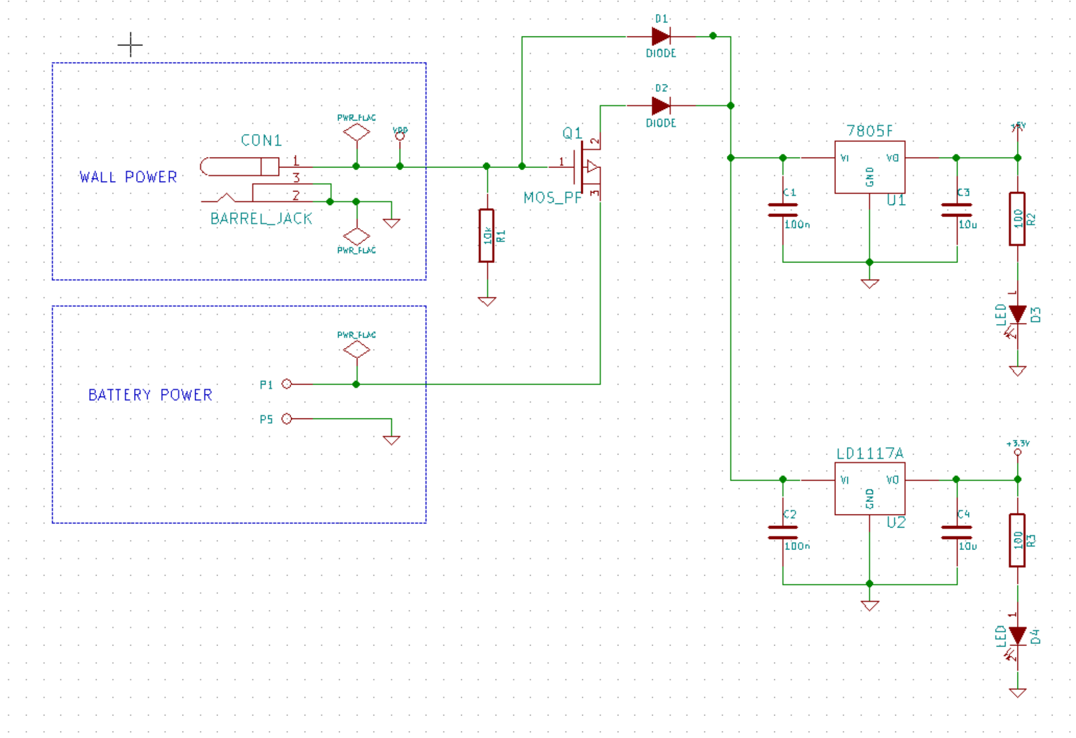


Figure 10 - Interface Board Power

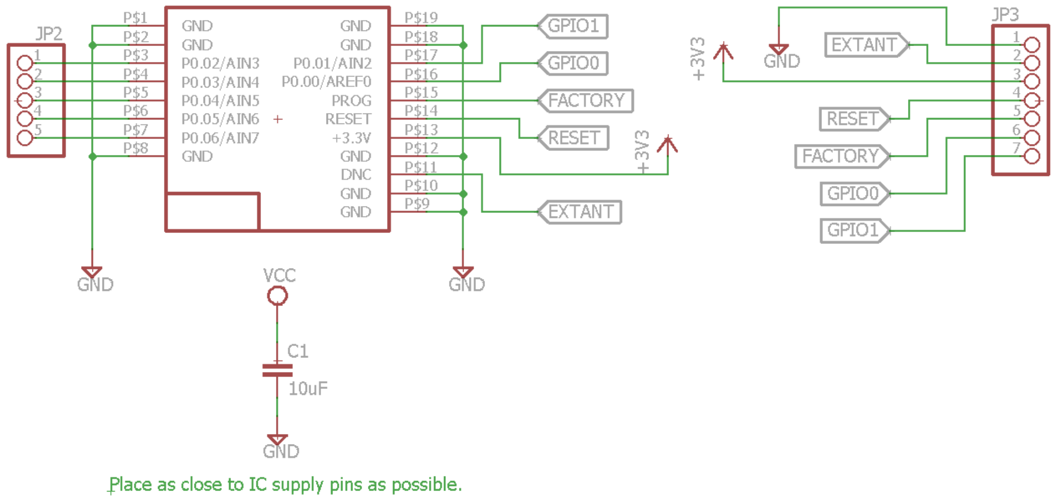


Figure 11 - Interface Board RFduino/Headers

The interface board power can come from two possible sources: DC power via a barrel jack, as well as from a 9 Volt battery. The battery is connected to the board via a p-channel MOSFET with the gate tied to the barrel jack. This ensures that the board will preferentially draw power from the wall when available, preserving the battery as a source of backup power.

These two power sources run to two different level voltage regulators, 5 V and 3.3 V. The 3.3 V will power the RFduino, as well as the low power sensors. The 5 V regulator will be used to power the CO sensor heater coil. Both of these lines are passed to the sensor boards via the headers on the interface board. JP2 and JP3 are arranged in a careful fashion to allow compatibility with the existing RFduino programming “shield”.

The entirety of the RFduino interface is connected to these headers to allow whatever signals may be necessary to be connected to the sensor boards. There is a single bypass capacitor that should be ceramic and placed as close as possible to the RFduino supply pins to allow delivery of fast current during high frequency switching periods. This capacitor is carried over from the example schematics posted by the RFduino developers.

4.3.1.1. Sensor Interface Header

The I/O header shown in Figure 9 is defined in Table 10 - Interface Board Pinout. The goal of the I/O header is to provide *all* available signals to the sensor, such that those that are needed are available and those that aren’t needed may be ignored.

Table 10 - Interface Board Pinout

Connector	Pin Number	Signal Description	Electrical Characteristics
JP1	1	5 V Power	5 V, 1 A
JP2	1	P0.02	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
	2	P0.03	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
	3	P0.04	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
	4	P0.05	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
	5	P0.06	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
JP3	1	Ground	0 V
	2	N/C (External Antenna)	N/A
	3	3.3 V Power	3.3 V, 1 A
	4	Reset	0 – 3.3 V
	5	Factory / Program	0 – 3.3 V
	6	P0.00	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA
	7	P0.01	I: 0 – 3.3 V O: 0 V, 3.3 V @ 5 mA

4.3.2. Carbon Monoxide

The CO sensor used in our project was the MQ-7, produced by Zhengzhou Winsen Electronics Technology. Electrochemical sensors were considered for the project, however these are not available for retail purchase, and are considerably more expensive. This sensor (the MQ-7) is a semiconductor-type detector, meaning that it requires heater power to operate, and presents its output as a variable resistance value, from which an ambient CO concentration can be determined by using the curve in the datasheet.

The sensor is sampled periodically, with the heater voltage set to 1.5 V when reading the sensor, after which 5 V is applied in order to refresh the sensor. The manufacturer suggests a high voltage time of 60 seconds followed by a low voltage period of 90 seconds. A simple timing circuit utilizing a 555 or similar IC could easily be constructed, but we chose to utilize the timers on the RFduino, and implement the controller in software, using a logic-level MOSFET in order to control the heater supply current.

The sensor resistance is read by measuring the voltage across a load resistor connected in series with the sensor resistance. The minimum load resistance (according to the manufacturer) is 5 kOhms, and the suggested load resistance is 10 kOhms (which will be the value used in our circuit). A voltage of 3.3 V (the logic level of the RFDuino, so as to avoid exceeding the ADC range and possibly damaging the microcontroller) is applied across both resistances. The sensor resistance is determined (from the voltage across the load resistor) as follows:

$$R_S = R_L\left(\frac{V_{CC} - V_L}{V_L}\right)$$

The schematic shown in Figure 12 indicates how the MQ7 was used in our sensor board. Note that the heater and sense circuits are depicted separately, since they are electrically separate. The n-channel MOSFET is used to control the voltage across the heater coil (shown here as a resistor). A PWM signal provided by the RFduio, is used to set the appropriate heater voltage during the low period of the sensor. PWM is adequate for this purpose since the response time of the heating coil is very long compared to the PWM period. Resistors RS (the sensor resistance) and RL (the 10 kOhm load resistor) form a voltage divider, feeding output to the RFDuino ADC, which yields a CO reading.

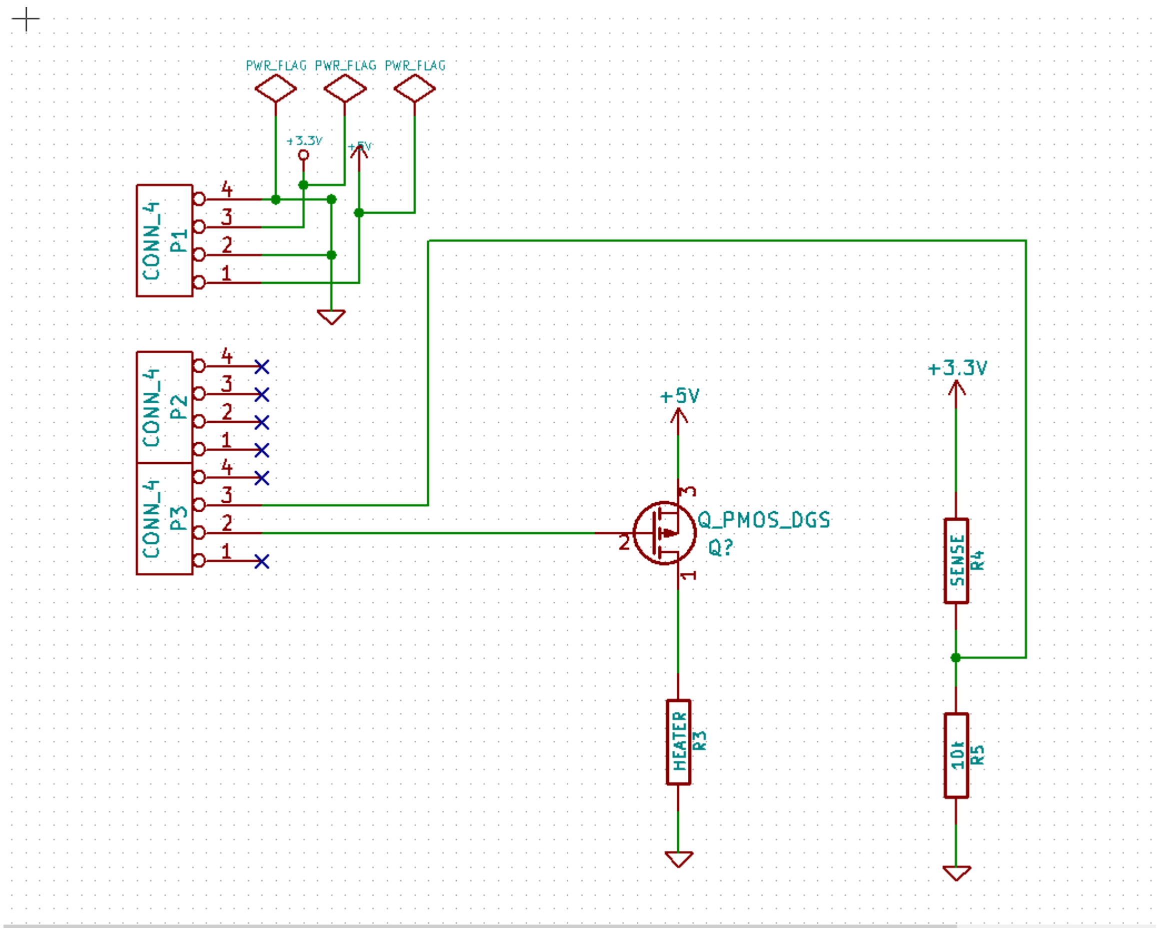


Figure 12 - CO Sensor Board Schematic

4.3.3. Humidity

The humidity sensor was the HS1100/1. This is a capacitive type humidity sensor which is targeted at high-volume, cost-sensitive applications, making it a good match for our project.

As this sensor is essentially a capacitor with a capacitance dependent on the humidity of the air it is in, means capacitance measurements (in some respect) are required in order to determine the humidity level being detected. A block-diagram of the approach to this problem is shown in Figure 13 - Humidity Sensor Interface Block Diagram.

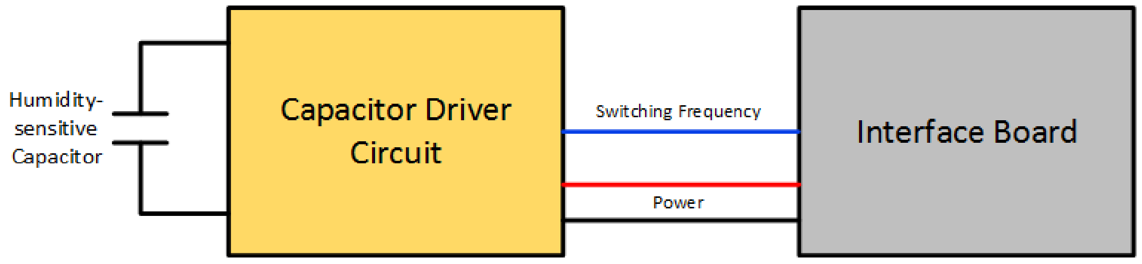


Figure 13 - Humidity Sensor Interface Block Diagram

The HS1100/1 datasheet provides a recommendation for such a capacitor driver circuit, as shown in Figure 14 - HS1100/1 Driver and Measurement Circuit.

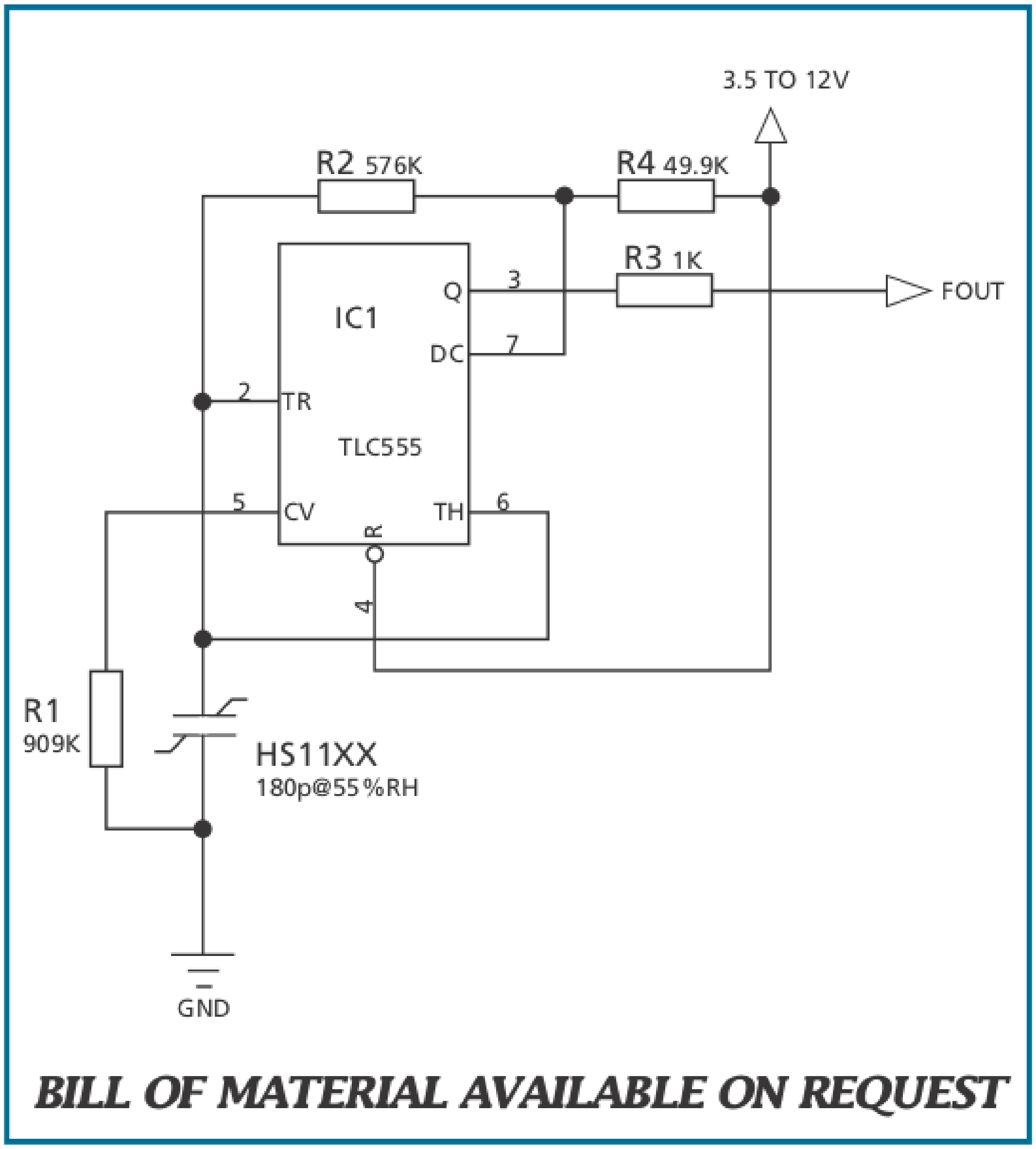


Figure 14 - HS1100/1 Driver and Measurement Circuit, used with permission from Parallax

This circuit repeatedly charges and discharges the humidity sensor capacitor, providing an output signal representing the frequency the capacitor is charging and discharging at. This value, dependent on the capacitance of the capacitor (more accurately, dependent on the time constant of the capacitor circuit. Thus the resistor R1 should be a 1% tolerance component to prevent it from influencing the time constant too much, as noted by the datasheet), thus provides an indication of the humidity of the air.

The TLC555 timer outputs a typical high level voltage of 2.85 V and a typical low level voltage of 0.07 V, making it suitable to connect directly to a GPIO pin of the RFduino, which can be configured to properly determine the frequency of output signal by use of timer peripherals, as detailed in section 5.1.2 Humidity Sensor.

4.3.4. Smoke

The smoke sensor consists of an infrared (IR) region LED, an IR region receiver, and a plastic housing. A system-level overview of the smoke detector is shown in Figure 15 - Smoke Detector Block Diagram.

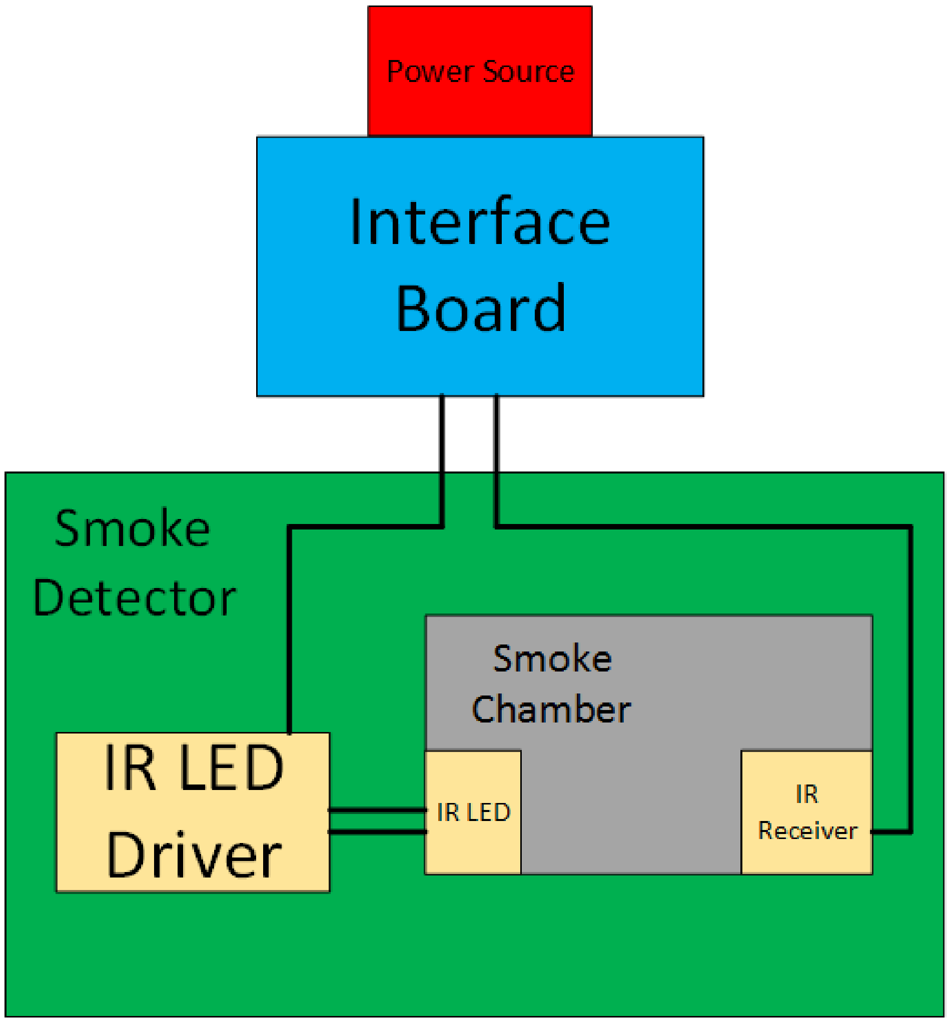


Figure 15 - Smoke Detector Block Diagram

The theory of operation is as follows the transmission of the IR led to receiver is normally inhibited by the housing. A simple "wall", coupled with the diffusive material of the housing, simply prevents the light from reaching the receiver. However, there is a channel to the side of the wall. When the chamber fills with smoke, the light is scattered by the smoke particles into the receiver, bypassing the wall [38].

A picture of this system is shown below in Figure 16 - Smoke Detection Chamber, followed by two images showing the theory of operation: Figure 17 - Smoke Chamber without Smoke and Figure 18 - Smoke Chamber with Smoke.

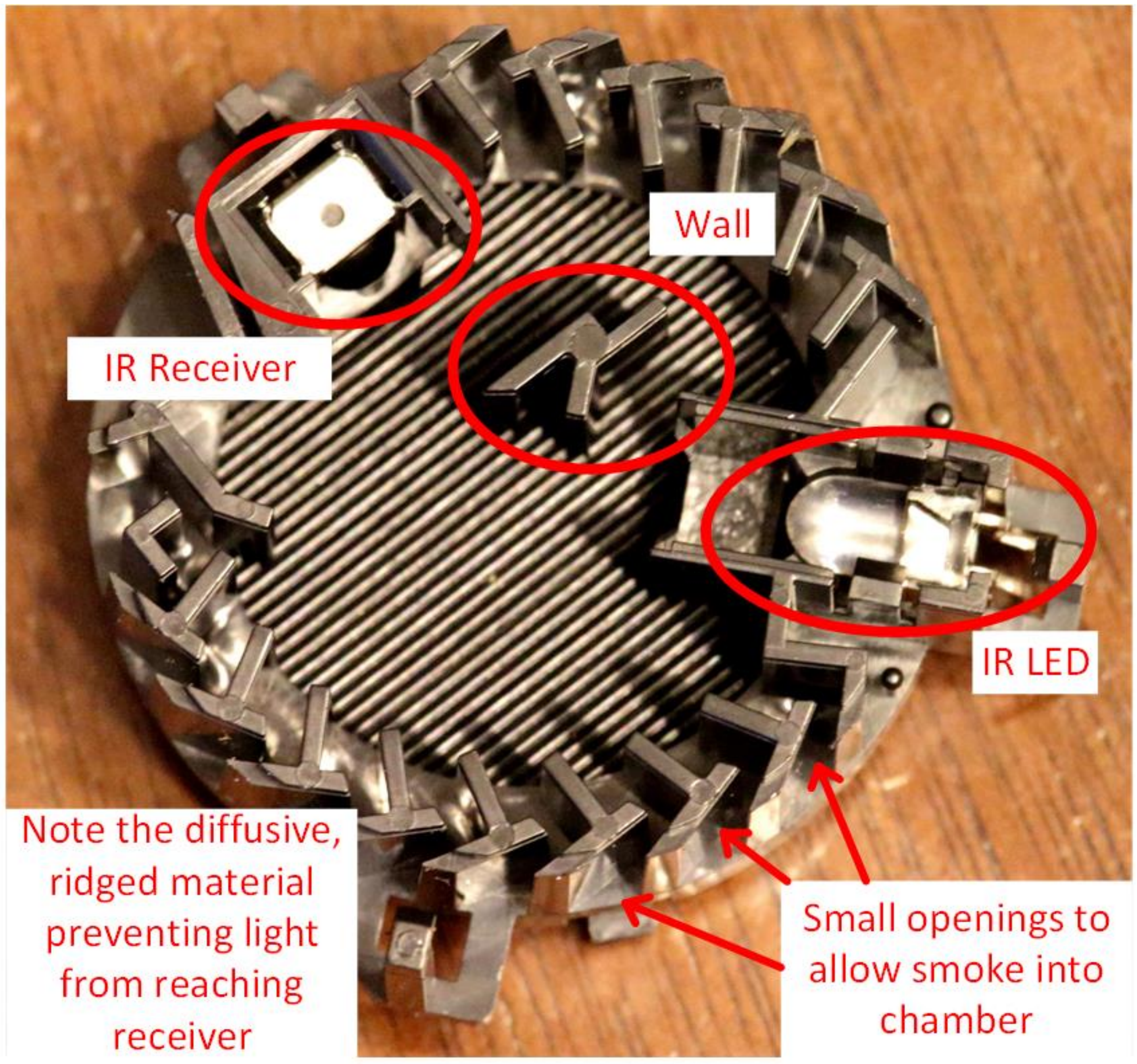

Wall
IR Receiver
IR LED
Note the diffusive, ridged material preventing light from reaching receiver
Small openings to allow smoke into chamber

Figure 16 - Smoke Detection Chamber

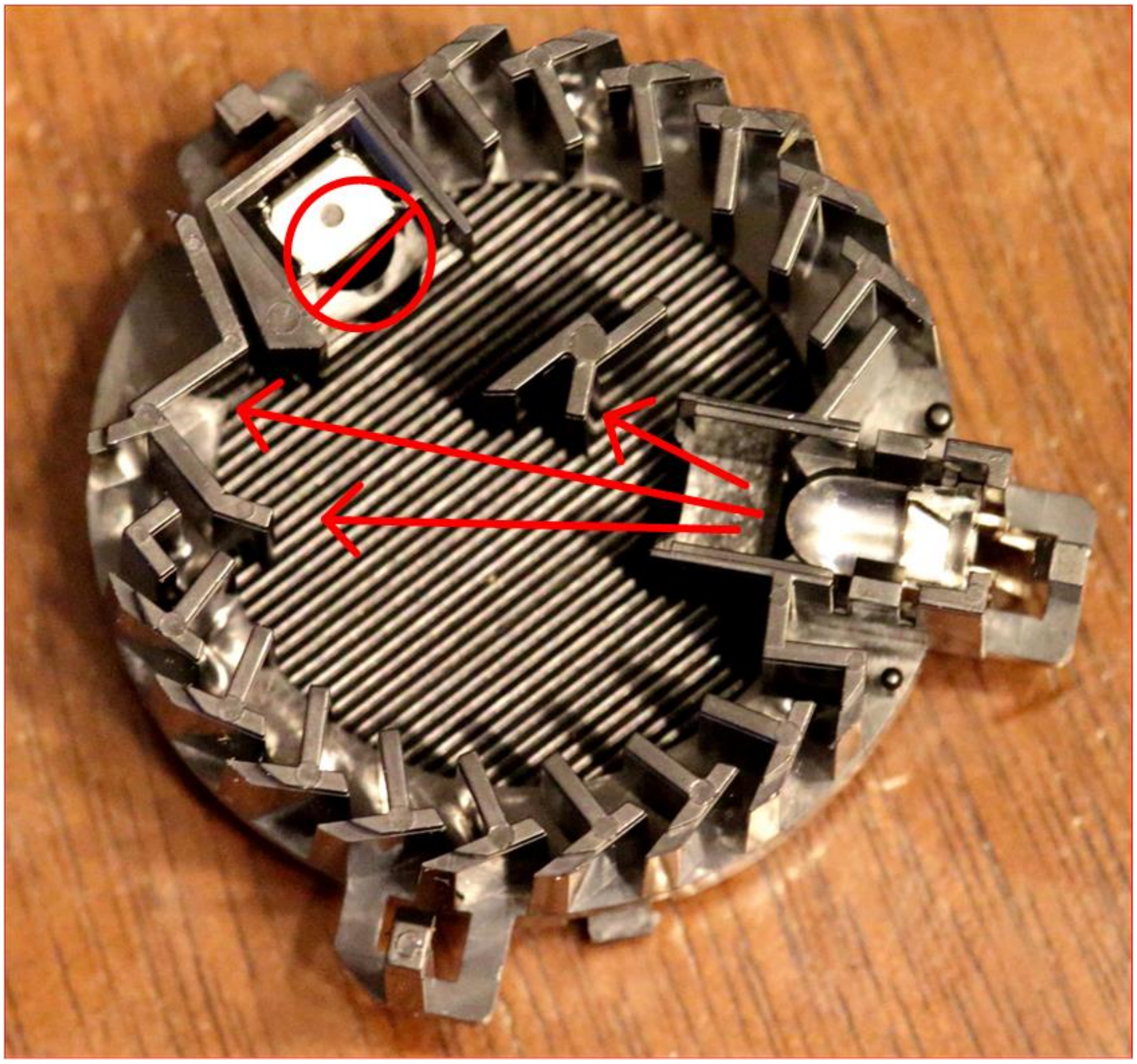


Figure 17 - Smoke Chamber without Smoke

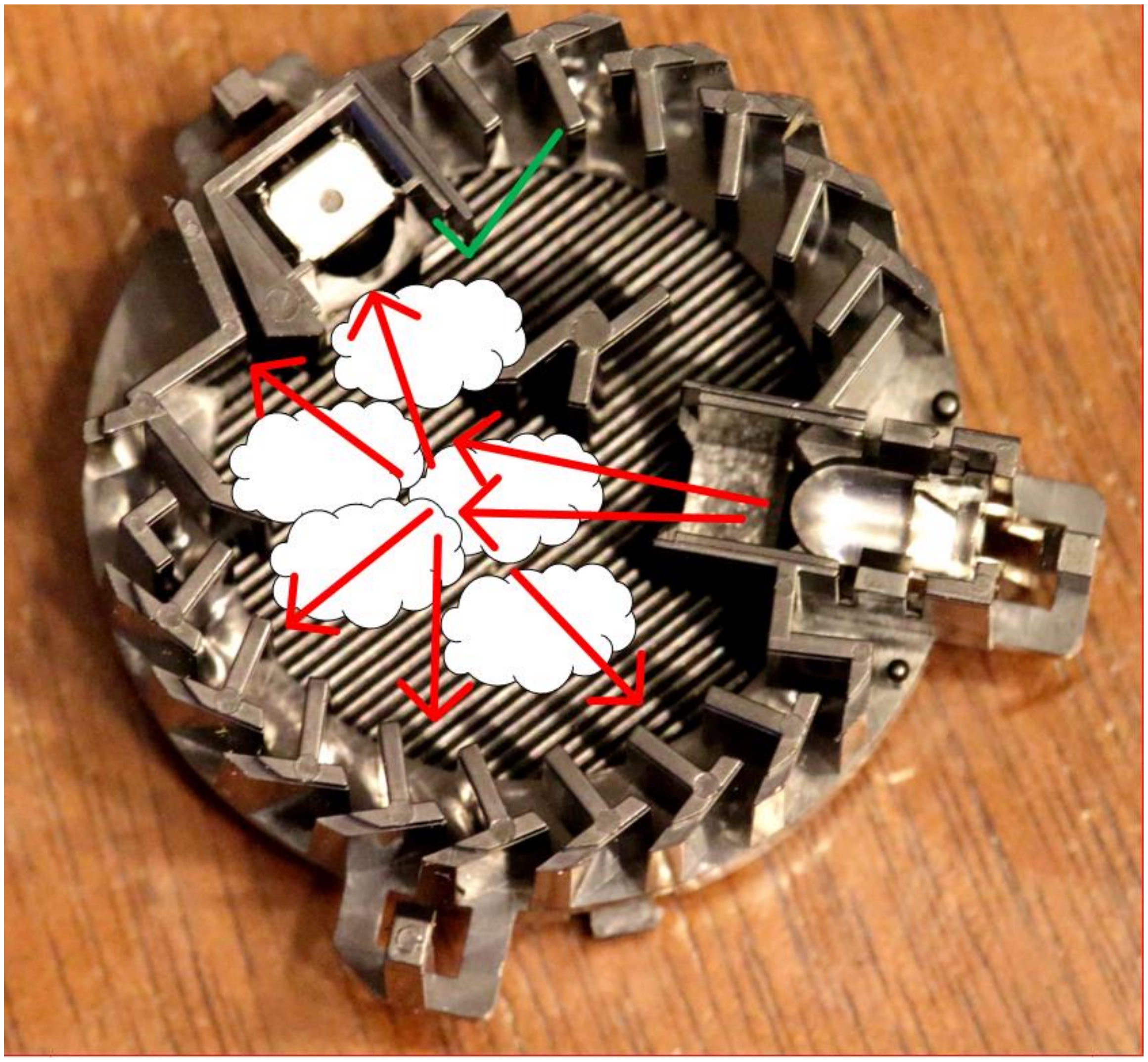


Figure 18 - Smoke Chamber with Smoke

It should be noted that this system will not detect fire or other heat sources which do not emit smoke – hence we refer to it only as a smoke detector, and not as a general fire or heat detection sensor.

The IR LED has a maximum current rating of 50 mA, and has a forward voltage drop in the range of 1.3 to 1.7 V. The RFduino GPIO pins have a maximum source current rating of 5mA. This is unsuitable for driving the IR LED directly. In addition, a continuous driving of the LED is unnecessary and would waste too much power to make battery operation feasible for an extended period of time. To save power, the LED will be pulsed quickly, but with a low overall duty cycle. This will allow accurate samples to be acquired every few seconds instead of continuously, which is unnecessary.

In order to drive the LED, a simple MOSFET driving circuit (which is inside the IR LED Driver block from the block diagram) will be used, and is captured in Figure 19 - IR LED Driving Circuit.

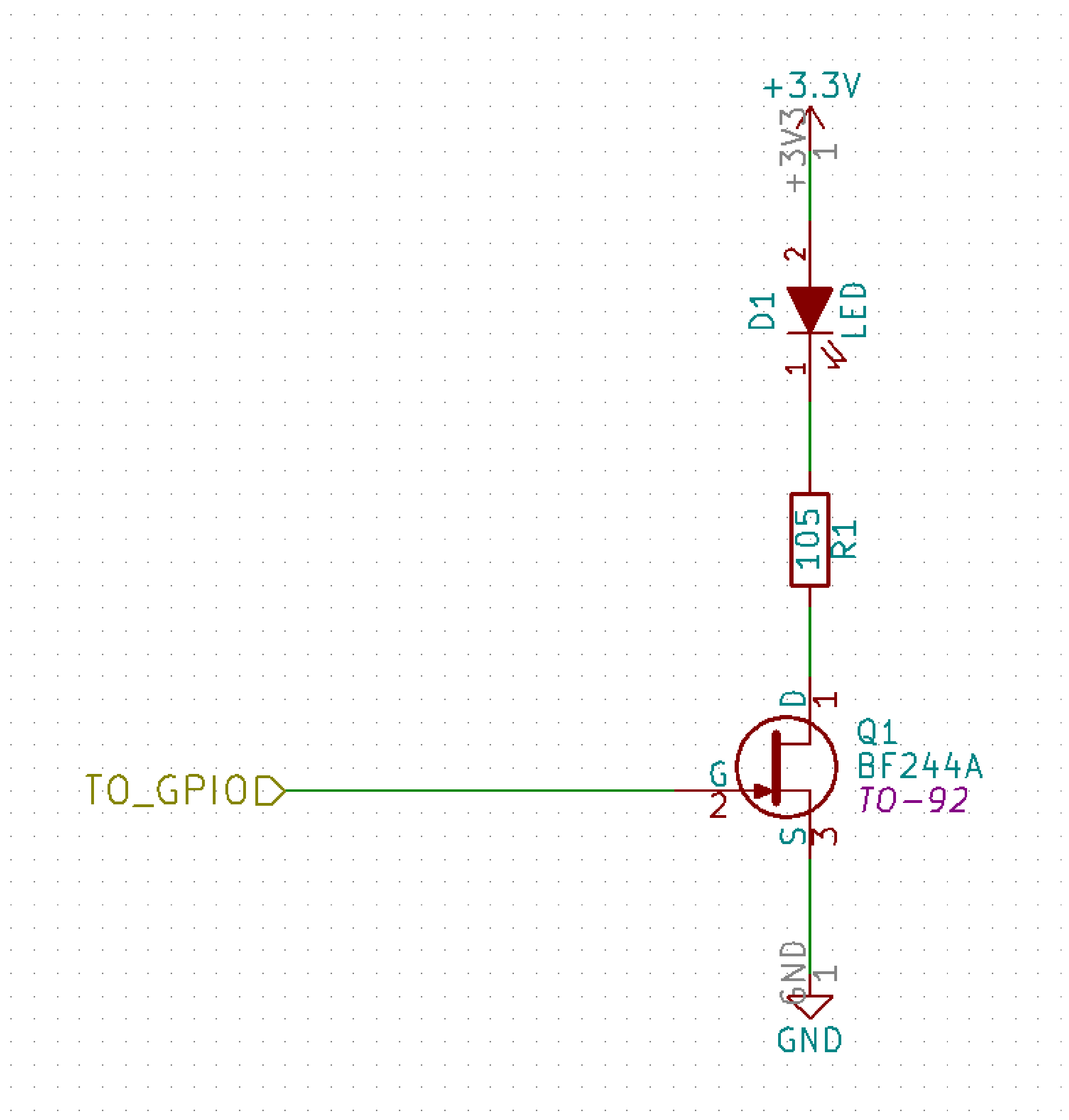


Figure 19 - IR LED Driving Circuit

This circuit is comprised of a MOSFET, a 105 Ω resistor, and the IR LED. Instead of trying to source the current to drive the IR LED (around 20 mA) from the RFduino with its limited driving capabilities (which would likely damage the pins), the power supply itself will provide the power.

The resistor value was selected as follows:

$$Power\ supply\ voltage = 3.3\ V$$

$$Forward\ voltage\ of\ LED\ = 1.2V\ to\ 1.7V$$

$$Desired\ current = 20\ mA$$

Given that the purpose of the resistor is to limit the current through the LED to prevent damaging it, it is desired that it limits the current to 20 mA over the entire range of the forward voltage of the LED. The worst-case condition is when the forward voltage is small as the current across the resistor will be higher. Thus, this voltage is used when calculating the value:

$$\frac{3.3\,V - 1.2\,V}{0.02\,A} = 105\,\Omega$$

The control functionality is maintained by the MOSFET – it is N-channel, and is easily turned on and off by the GPIO pin of the RFduino. A MOSFET with below a 2-3 V gate-source threshold voltage should be used to ensure the MOSFET is turned on hard enough to act as a low-resistance connection to the ground.

4.4. Camera

The camera used was the Raspberry Pi Camera Board. This camera board mates directly with the CSI port of the Raspberry Pi / Main Control Unit itself. It features a flexible ribbon cable which allows limited movement of the camera in order to be positioned in a desirable manner.

Its features include the following:

- Small physical size (25 mm x 20 mm x 9 mm)
- 5 MP (2592 x 1944 pixels) image sensor
- Supports 1080p30, 720p60, and 640x480p60/90 video
- Existing software libraries for interfacing

As a result of the camera being ready to connect out of the box, the only hardware concerns regarding the camera were mounting and securing it. An existing commercial Raspberry Pi case will be used to house the Raspberry Pi / Main Control Unit, and it features a mounting location for such a camera. This mounting location will be utilized, and the Raspberry Pi as a whole will be oriented as necessary to provide the camera with the optimal field of view.

5. Software Design

5.1. Sensor Software

The MHMS uses an RFduino Bluetooth enabled microcontroller for all logic and communications on the Sensor Pod's Interface Board. The RFduino is a fully featured Arduino chip, thus it will be programed through the Arduino IDE using the C language.

The software on the RFduino is responsible for connecting to the MCU, performing its measurements, making decisions on sending alert messages and sending periodic status messages. It was decided that the RFduino should be responsible for managing these things instead of the MCU having to ask the RFduino for these actions to be performed in order to cut the amount of Tx and Rx time required.

Special consideration was taken to design software that consumes limited power for the Smoke and Humidity sensor pods since they should be capable of running completely on battery power. CO sensor software has the luxury of being designed to be powered by wall power because of its sensor, so power saving is not as much of a concern. While all the types of sensor pods will have the same Interface Board, because of the differences in the actual sensors, the software is slightly different for each type of pod. The software for the different sensor pods follow the following flow chart diagrams. These are shown in the sensor's respective subsections below.

5.1.1. CO Sensor

The CO sensor pod is the black sheep of the sensor pods because it consumes much more power than the other sensors. The software flowchart in the following figure shows how the software will be designed.

CO Sensor Pod Software Flowchart

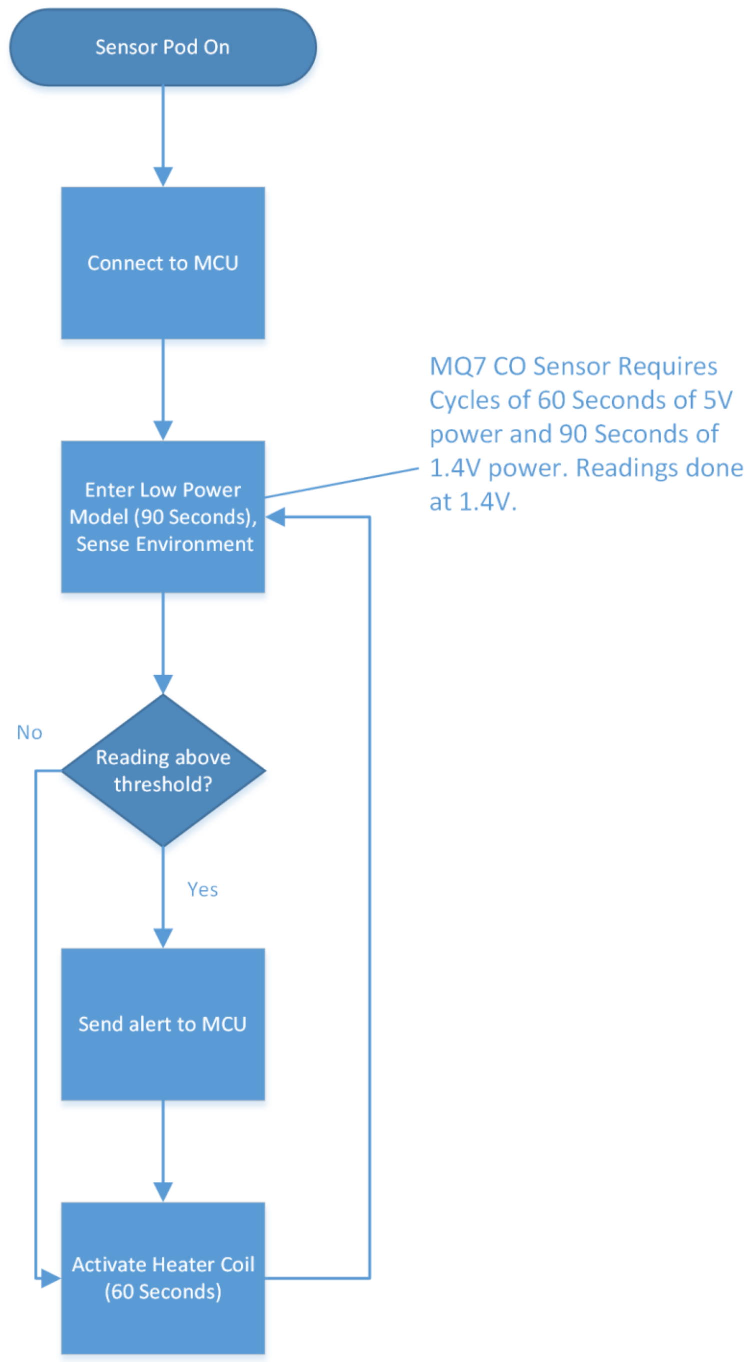

Sensor Pod On
Connect to MCU
MQ7 CO Sensor Requires Cycles of 60 Seconds of 5V power and 90 Seconds of 1.4V power. Readings done at 1.4V.
Enter Low Power Model (90 Seconds), Sense Environment
No
Reading above threshold?
Yes
Send alert to MCU
Activate Heater Coil (60 Seconds)

Figure 20 - CO Sensor Pod Software Flowchart

5.1.2. Humidity Sensor

The humidity sensor outputs a signal whose frequency corresponds to a humidity level. It will be the job of the RFduino to process this signal, determine its frequency, and translate that frequency to a humidity level. This process is shown in Figure 21 - Humidity Sensor Pod Software Flowchart.

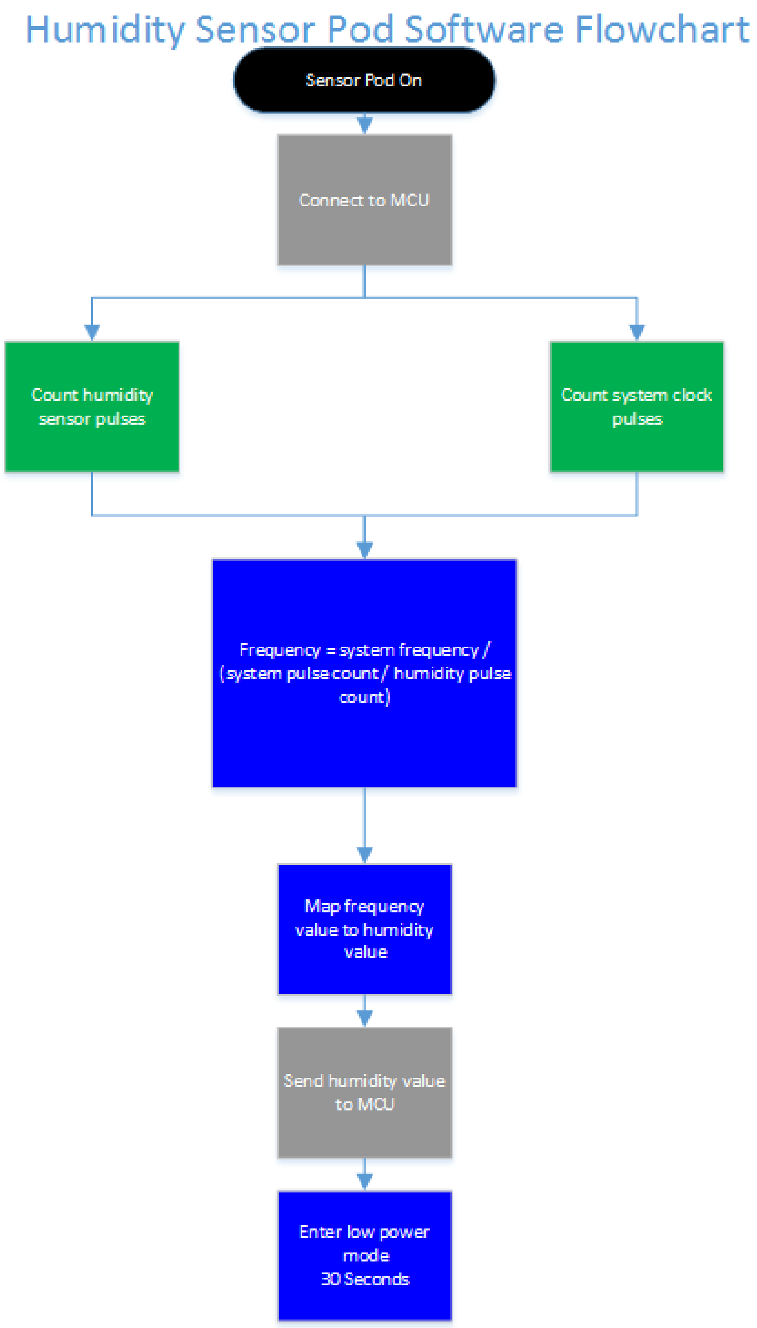

Humidity Sensor Pod Software Flowchart
Sensor Pod On
Connect to MCU
Count humidity sensor pulses
Count system clock pulses
Frequency = system frequency / (system pulse count / humidity pulse count)
Map frequency value to humidity value
Send humidity value to MCU
Enter low power mode 30 Seconds

Figure 21 - Humidity Sensor Pod Software Flowchart

This method of determining the frequency of a signal is simple, and timer peripherals that exist in hardware as part of the RFduino can be employed as an efficient means of counting signal pulses.

The method uses a comparison between a known, much higher frequency and the unknown frequency. It is important that the known frequency be sufficiently higher in value than the unknown frequency – if it is not, you will suffer from aliasing the signal to the point of complete error.

The expected frequency of oscillation from the hardware capacitor driver will be on the order of kilohertz. The system clock frequency of the RFduino is on the order of megahertz, resulting in a 3 order of magnitude difference which will entirely eliminate the possibility of aliasing the signal. One must sample at more than twice the rate of the highest frequency content (that is desired to be measured) of a signal in order to avoid any information loss [39], and our system will easily exceed that.

A counter will continuously count system clock pulses (or some sufficiently high frequency, derived signal's pulses), until a humidity sensor pulse arrives. At this time, the system will determine how many system clock pulses occurred before the humidity sensor pulse, and can thus determine the unknown humidity sensor pulse frequency by simple division of the higher, known, frequency. A timing diagram of this functionality is shown in Figure 22 - Unknown Frequency Determination.

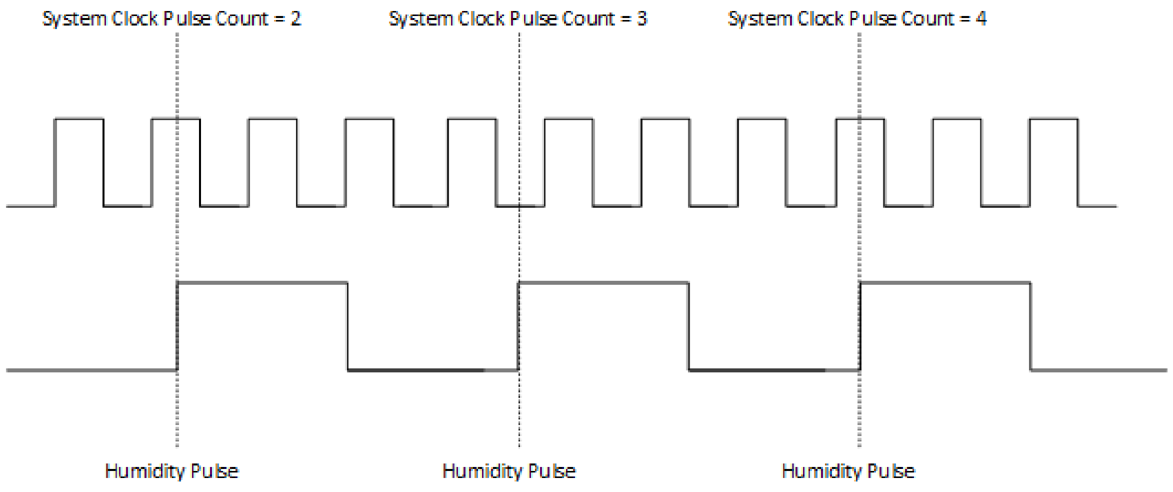

System Clock Pulse Count = 2
System Clock Pulse Count = 3
System Clock Pulse Count = 4
Humidity Pulse
Humidity Pulse
Humidity Pulse

Figure 22 - Unknown Frequency Determination

A single sample of this method could have significant error, depending on the variability of the asynchronous relationship between the system clock and the humidity sensor output. To reduce this error, the method can be repeated any number of times, allowing for averaging or other statistical operations to bring the value closer to the actual value.

After the frequency value has been accurately determined, it must be interpreted. The output frequency varies linearly with RH, as shown by a plot of data from the HS1100/1 datasheet in Figure 23 - Output Frequency vs. RH.

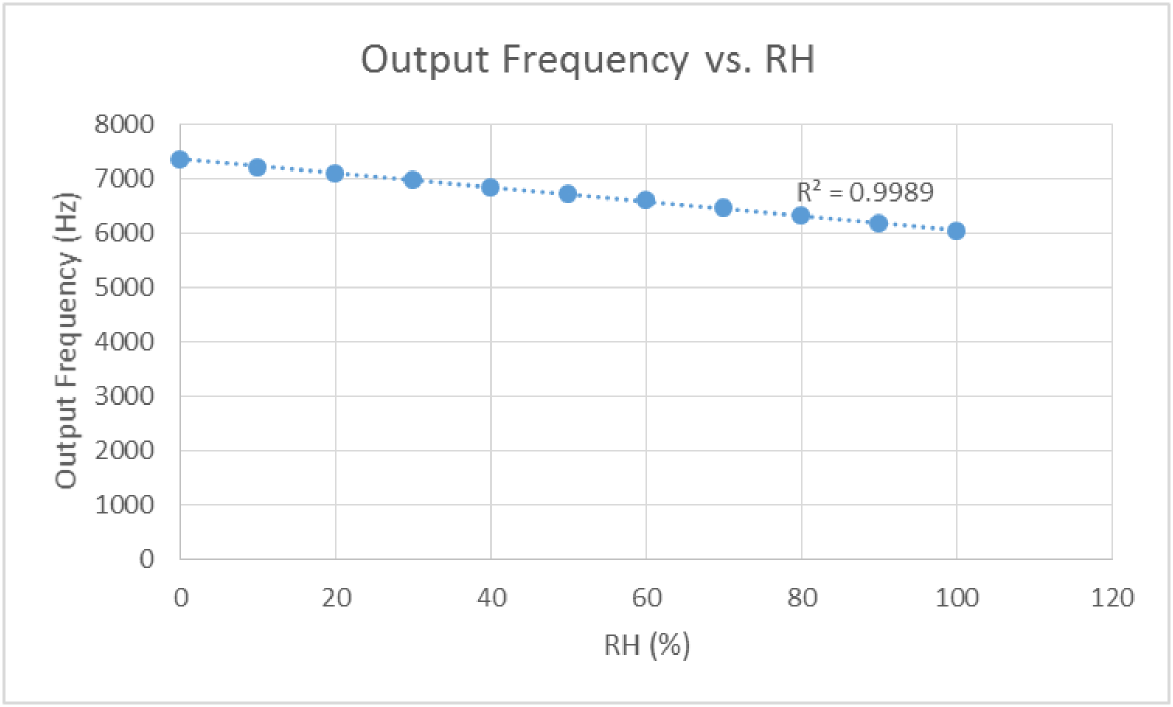


Figure 23 - Output Frequency vs. RH

It can be seen that all that is needed is a nominal reference point, with all deviations from the reference point interpreted in a linear fashion, where higher output frequencies correspond to a lower relative humidity, and vice versa. It can also be seen by the accompanying R^2 value that a linear interpretation of this data is quite acceptable.

The actual response of the sensor to RH is also given in the datasheet, and is reproduced here for reference in Figure 24 - Humidity Sensor Response Equation. Note the relatively small sizes of the nonlinear term coefficients – further justifying the use of linearization for interpretation.

Polynomial response :

$$F_{mes(Hz)} = F55_{(Hz)}(1.1038-1.936810^{-3}*RH+3.011410^{-6}*RH^{2}-3.440310^{-8}*RH^{3})$$

Figure 24 - Humidity Sensor Response Equation

5.1.3. Smoke Sensor

The smoke sensor software is tricky as this is one that the user will most likely worry about most. Because of this, we want to ensure the software is capable enough to not alert on false positives. To do this this software will do a wait and check again to make sure we are seeing smoke in the chamber. This is similar to how real smoke detectors work in the market.

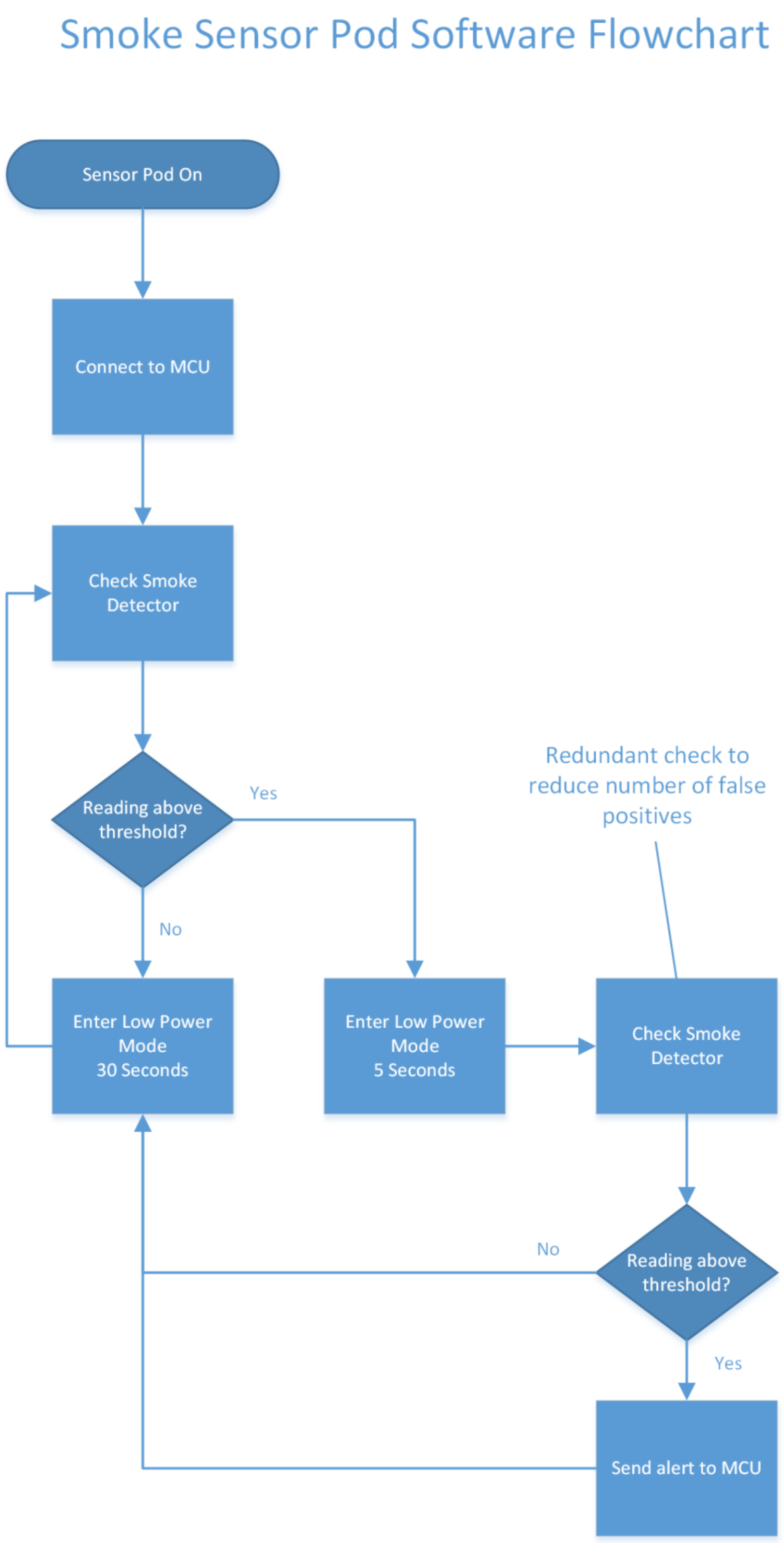

Smoke Sensor Pod Software Flowchart
Sensor Pod On
Connect to MCU
Check Smoke Detector
Reading above threshold?
Yes
No
Redundant check to reduce number of false positives
Enter Low Power Mode 30 Seconds
Enter Low Power Mode 5 Seconds
Check Smoke Detector
No
Reading above threshold?
Yes
Send alert to MCU

Figure 25 - Smoke Sensor Pod Software Flowchart

5.2. Communication

The format of the data sent from the RFduino to the MCU and vise-versa is especially important. Because we are using Bluetooth Low Energy (BLE) IBeacon communications provided by the RFduino we did not have to worry about the headers and the specifics on how the data is moved between the RFduino and the MCU, but we did have to come up with a format for our data that will fit in the IBeacon's 16 bit major and minor payload fields. Because of the constraints given to us by the protocol we are following, we had to go with a 32 bit binary data format that includes the following fields:

16 bits (Major)	16 bits (Minor)
Sensor Data	**Sensor Type**

The fields will be defined according to the following definitions:

Field Name	**Defined Values**
Sensor Type	0 Reserved 1 Smoke Sensor Pod 2 CO Sensor Pod 3 Humidity Sensor Pod 4 – 37,767 Undefined
Sensor Data	0x0000000000000000 – 0xFFFFFFFFFFFFFFFF Possible values for sensor readouts

Table 11 - Defined Message Data Field Values

5.3. Main Control Unit Software

The MCU acts as a middle man for moving data between the sensors and the cloud application and it monitors the broadcast of each Sensor pod connected to it. On top of that, the MCU is responsible for capturing frames from the camera and sending it to the web user interface at least once per second on command. The MCU software includes a few processes that take care of these tasks.

The process that controls communication has two important tasks. First, it scans the area around it for broadcasting beacons. It is possible that the user will introduce a new sensor pod to the network at any given moment, so the software is able to quickly pick up the addition of new beacons on the Bluetooth network. Every beacon scan cycle, the MCU extracts the Address, UUID, Major and Minor fields from every beacon it sees. Since it is not enough to do just one scan then send, several scans are performed before data is sent to the Cloud to prevent a non-continuous transmitting beacon from being ignored. When the beacon scan cycle is complete, the software packs the data into a message for the cloud service and sends it off via a client API. The cloud application is then responsible for interpreting and displaying data to the user including status of sensors, alerts and potentially sensor readouts.

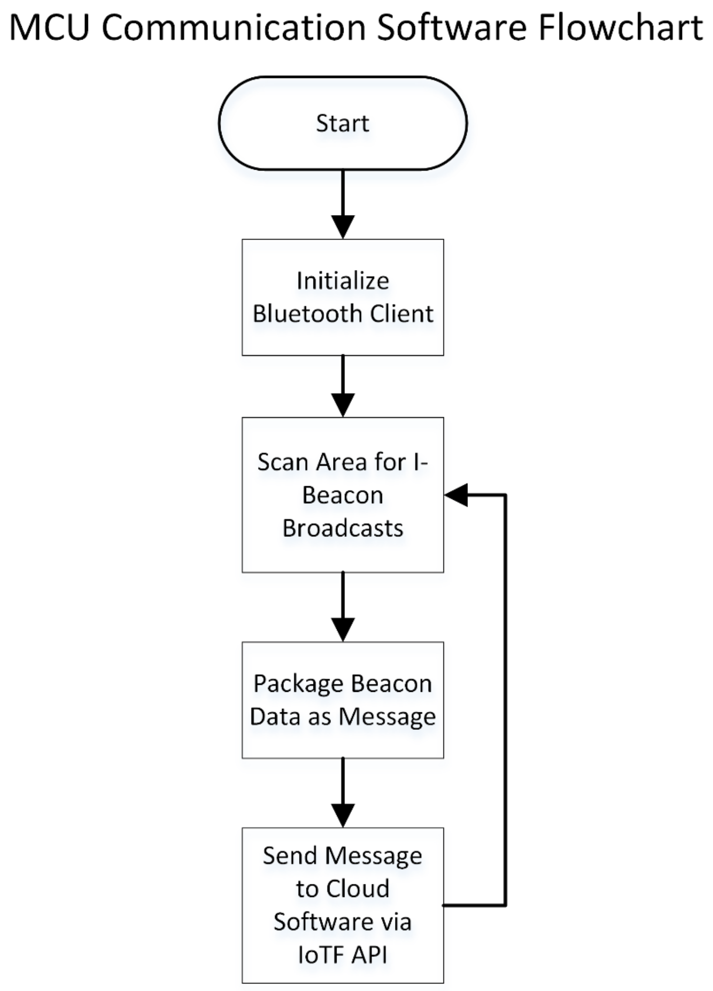

MCU Communication Software Flowchart
Start
Initialize Bluetooth Client
Scan Area for I-Beacon Broadcasts
Package Beacon Data as Message
Send Message to Cloud Software via IoTF API

Figure 26 - MCU Communication Flowchart

5.3.1. Camera

To stream the series of still images from the Raspberry Pi's Camera to the Web User Interface, the Main Control Unit uses a custom built software package to take, encode, send and decode pictures. The software package has the capability of transmitting an image stream to the Cloud at a specified frame rate and resolution, however because of limitations on data usage, care was taken to ensure the image stream is not always transmitting. The stream is only be active when the user is logged into the user interface and clicks a button to start the steam. That button sends a signal to the MCU to start sending images. The images taken on the MCU are encoded to base-64 in order to be split into packets and sent to the cloud server via MQTT. From the cloud server, a MQTT client on the web page receives the packets, decodes the image and displays it on the web page. A flowchart of this process is shown in in Figure 27.

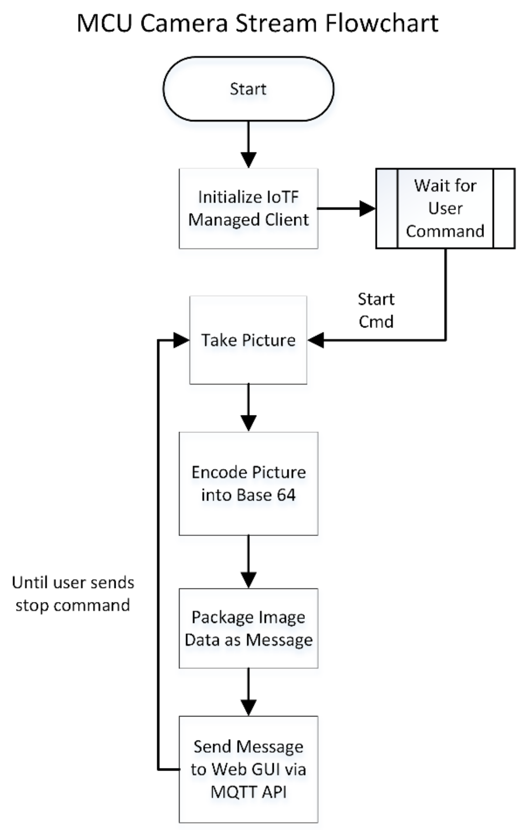

MCU Camera Stream Flowchart
Start
Initialize IoTF Managed Client
Wait for User Command
Start Cmd
Take Picture
Encode Picture into Base 64
Package Image Data as Message
Send Message to Web GUI via MQTT API
Until user sends stop command

Figure 27 – Camera Streamer Flowchart

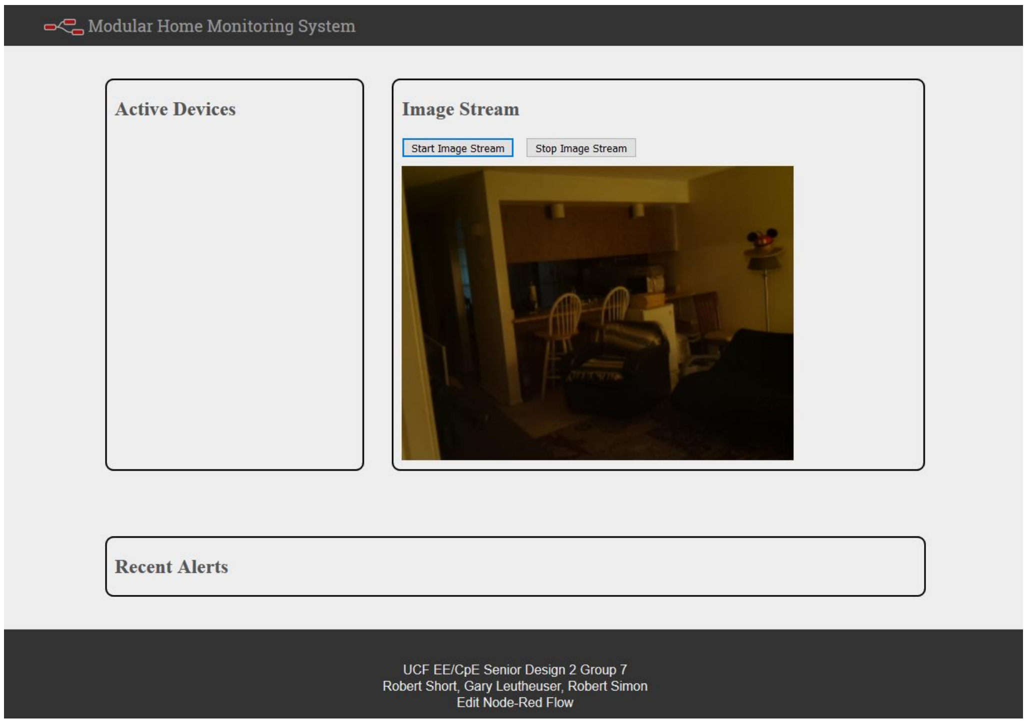

Modular Home Monitoring System
Active Devices
Image Stream
Start Image Stream
Stop Image Stream
Recent Alerts
UCF EE/CpE Senior Design 2 Group 7
Robert Short, Gary Leutheuser, Robert Simon
Edit Node-Red Flow

Figure 28 - Camera Stream Working on GUI

5.4. Cloud Application

The cloud application was decided to run on a cloud PaaS provider that allows for hosting a web page, Internet of Things (IoT) connectivity and database management. IBM Bluemix is the PaaS that the MHMS uses for these services. The bulk of the cloud application takes the form of a node-red flow that performs logic functions and routing on messages received from the Internet of Things Foundation connection with the MCU. The node-red flow can be seen in Figure 29.

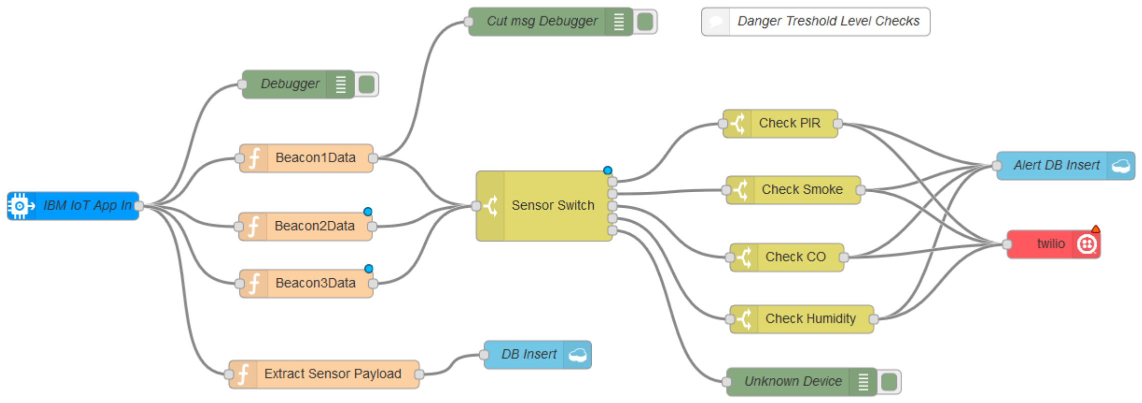


Figure 29 - Cloud Application Node Red Flow

The MCU sends all sensor data to the cloud application without doing any interpretation of data. The application decodes the message to determine the sensor type and content. Once it has this information, it is stored in a non-relational IBM cloudant database called Sensor Status. If the readings are higher than a certain threshold, the message is sent to a Sensor Alerts database and an alert is displayed on the web user interface's recent alerts panel. Using a simple database check, the system checks if this is an alert that has not been sent to the user already, if not, a text message is sent to the user via a web SMS provider. The SMS provider used by the MHMS is Twillio due to its low cost and ease of use. The sequence diagram in Figure 41 describes the system cloud software.

We have decided to use a non-relational database as our data store because of the small variety of data we actually have to store and the fact that non-relational DB's are easier to learn how to use. Our database only stores information about alerts and status messages from the sensor pods and such it will not be very complex. For better understanding, a tabular diagram is shown of what the database will actually store in Figure 31.

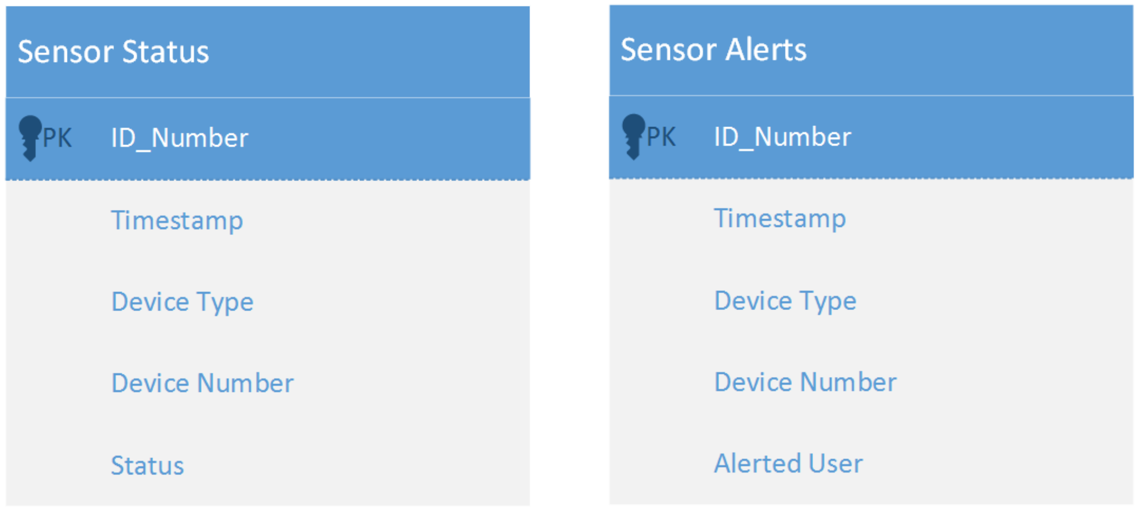


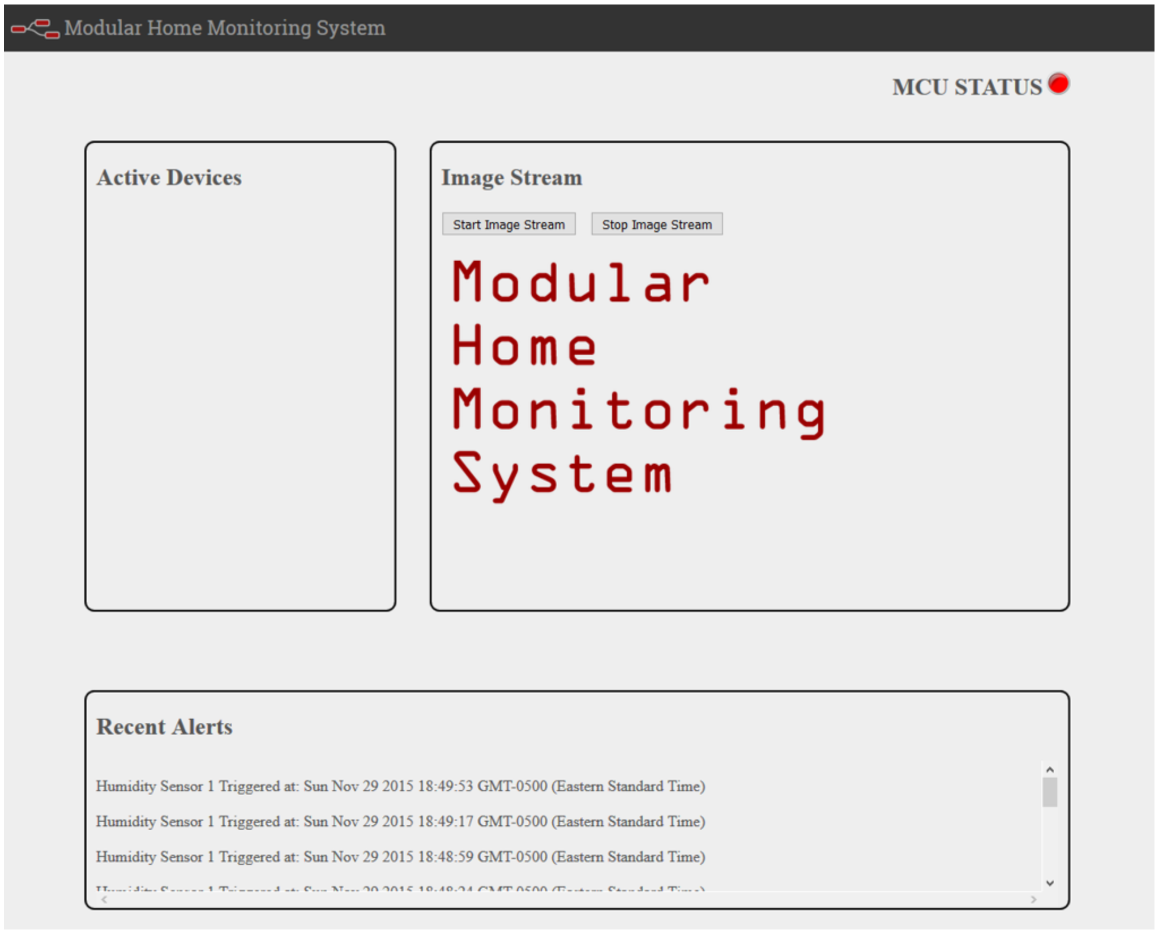
Figure 31 - Database Tables

5.5. User Interface

Any good home monitoring or automation system must have a way to either interact with or display data to the user. This is the whole point systems like these, thus the User Interface is a very important component of the overall system. For the MHMS, it was debated whether the interface should be a program that runs on the user’s computer, or a web interface that can be loaded on any web browser. Our group decided on using a web based user interface accessible and fully featured even when the user is not within their home network.

The user interface of the MHMS is a web with access to information on the status of your connected Sensor Pods, recent alerts, and a stream from the MCU Camera. The sensor pod status section of the page shows if the pods are active and sending periodic messages or if they are reporting some issue that needs attention. A client side script checks the database for new alerts and display the alerts as well as a timestamp in the recent alerts panel. The image stream from the MCU camera is displayed on the right hand side of the user interface to provide users the ability to check on their home visually even when they are not there. Two buttons above the stream allow the user to start and stop the image stream, however if the user leaves the page some other way the stream will automatically stop to conserve resources. A Status indicator informs the user if the MCU is online as well.

A good user interface must also be responsive and have good aesthetics, which is why the MHMS user interface has a modern style and scalable design that uses pre-existing styling libraries. The user interface pulls data about sensors and alerts from a database every 5 seconds using a script. Simple database calls are used within the script to retrieve any updated information to be displayed. Because it is bad practice to have to reload the entire page when information needs to be updated, the User Interface is designed in such a way that only the parts that need to be changed will be updated. A screenshot of the user interface can be found in Figure 32.



Humidity Sensor 1 Triggered at: Sun Nov 29 2015 18:49:53 GMT-0500 (Eastern Standard Time)
Humidity Sensor 1 Triggered at: Sun Nov 29 2015 18:49:17 GMT-0500 (Eastern Standard Time)
Humidity Sensor 1 Triggered at: Sun Nov 29 2015 18:48:59 GMT-0500 (Eastern Standard Time)

Figure 32 - User Interface

6. Prototype Construction

6.1. Bill of Materials and Part Acquisition

The total Bill of Materials can be found in the table below. This bill includes all materials to produce one CO, one Smoke and one Humidity sensor pod as well as the Main Control Unit and camera.

Line #	QTY	Part Description	Vendor	Vendor Part Number
1	1	Raspberry Pi 2 B	Allied Electronics	70465426
2	1	Raspberry Pi 2 Case (Clear)	Amazon/ STC Components	B00MQLB1N6
3	1	Raspberry Pi 2 Camera Module	Allied Electronics	70280250

4	1	Edimax EW-7811Un 11n Wi-Fi USB Adapter	Amazon/ HometownTV plus	B003MTTJOY
5	1	Plugable USB Bluetooth 4.0 Micro Adapter	Amazon/Plugable Technologies	B009ZIILLI
6	1	NorthPada USA Micro USB 5V/2A Power Wall Supply	Amazon/Amboc	B00OY7HR1U
7	3	RFDUINO BLE SMT	Semiconductor Store	RFD22301
8	1	RFDUINO USB Programming Shield	Semiconductor Store	RFD22121
9	1	Raspberry Pi Camera Case	Amazon/SB Components	B00IJZJKK4
10	1	DC Power Jack	Mouser	502-RAPC712X
11	1	Coin cell battery holder	Mouser	712-BAT-HLD-001
12	1	Female headers	Sparkfun	11269
13	2	Rectifier diode	Mouser	625-P600M-E3
14	1	5 V Linear Regulator	Mouser	863-NCP7805TG
15	1	3.3 V Linear Regulator	Mouser	511-LD1117AV33
16	3	100 nF Capacitor	Mouser	647-UVZ2A0R1MDD
17	1	10 µF Capacitor	Mouser	667-ECE-A1VKS100
18	1	75 Ω Resistor	Mouser	594-SFR16S0007509FR5
19	1	150 Ω Resistor	Mouser	652-WS2M1500J
20	2	Red LED	Mouser	667-LN28RPX
21	1	10 uF Capacitor	Mouser	810-FK18X5R0J106M

6.2. Facilities and Equipment

Facilities and equipment are a critical practical element requiring consideration in order to meet the design requirements within the scheduled time period. Facilities and equipment were considered on the basis of cost and availability. The following facilities were used throughout this project:

- UCF Senior Design Lab
- UCF TI Innovation Lab
- UCF Idea Lab
- Group member's home and garage

A list of the equipment that were used in the implementation of this project are shown in Table 12 - List of Equipment.

Table 12 - List of Equipment

Equipment	Model Number	Facility
Oscilloscope	• Rigol DS1102E • Tektronix MSO 4034B	• Home/Garage • UCF TI Innovation Lab • UCF Senior Design Lab
Digital Multimeter	• Tektronix DMM 4050 • Craftsman 82141	• Home/Garage • UCF TI Innovation Lab • UCF Senior Design Lab
DC Power Supply	• Agilent E3630A	• UCF TI Innovation Lab • UCF Senior Design Lab
Soldering Station	• Hakko Fx888	• Home/Garage • TI Innovation Lab
Function Generator	• Tektronix AFG 3022	• UCF TI Innovation Lab • UCF Senior Design Lab
PC	• Custom, AMD X4 750K • Dell Optiplex 960	• Home/Garage • UCF TI Innovation Lab • UCF Senior Design Lab
Workbench	• N/A	• Home/Garage • UCF TI Innovation Lab • UCF Senior Design Lab
Whiteboards	• N/A	• Home/Garage • UCF Idea Lab • UCF Senior Design Lab

Assorted tools (screwdriver, drill, etc.)	• N/A	• Home/Garage

6.3. Physical Packaging

The following sections describe how the Modular Home Monitoring System prototype is physically packaged, including how the electronics are housed in a physical enclosure.

6.3.1. Sensor Pods

All three sensor pods must be housed in an enclosure that protects against dust, yet allows for the measurable quantities in the air to enter the enclosure to be measured. The enclosures must also not be a huge eye-sore since they are designed to be around a user's home. However, because the focus of the prototype was to show a functioning version of the MHMS, the housing's looks were not a high priority for the group. The sensor pod enclosures are be box-shaped, with the top and sides being made of cut Plexiglas to allow for access to the power cables and ventilation so smoke, CO, or humid air can enter the enclosure for the sensors to measure. A hinge will allow for the housing to be opened for access to the whole pod circuitry. Four short 5mm sendoffs were installed in the enclosure to prevent the PCB and other circuitry from touching the wood.

Special consideration was made for the RF antenna included on the RFduino chip on the interface board. Because Bluetooth uses RF signals, the placement of the chip on the board and also the board in the enclosure and environment is very important. The RFduino chip must be as close as possible to the edge of any enclosure it is in. On top of that there must be as much free air around the enclosure where the RFduino is as possible to have optimal signal range. Because metal blocks RF signals drastically, there is an area of space void of metal objects for the RF signals to propagate. In their data sheet, the RFduino Company gives an example.

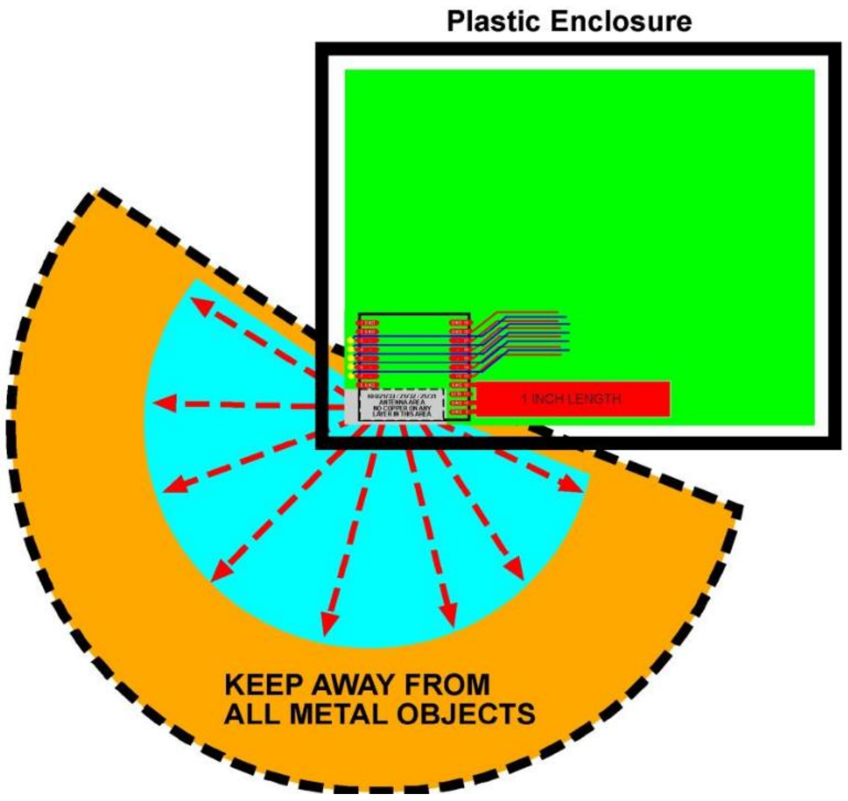


Figure 33 - RFduino Keep out area example (Used with permission from the RFduino Co.)

Due to the multiple needs outlined previously surrounding the sensor pod enclosures, we consulted with a student experienced in 3D model construction and procured a 3D model of our concept of a sensor pod enclosure. The enclosure features four jack posts which can secure any protoboard or PCB in place while the enclosure is mounted in a variety of configurations (be it on a ceiling or wall). The enclosure also features a hole in the side in order to run cabling out when needed (when using main power).

In addition, the small form factor and clean appearance are desirable aesthetic elements. In addition, LED power indicators can be clearly seen through the Plexiglas cover. This cover is on a hinge to allow the PCBs and other electronics to be easily accessed, tested, and maintained.

Finally, the Plexiglas features cutouts where the sensor is mounted to allow enough airflow to subject the sensors to the actual conditions the enclosure is located in – this will allow the CO, smoke, and humidity sensors to actually detect what they are designed to detect. Too much isolation would result in sluggish (or nonexistent) sensor reactions to changing conditions.

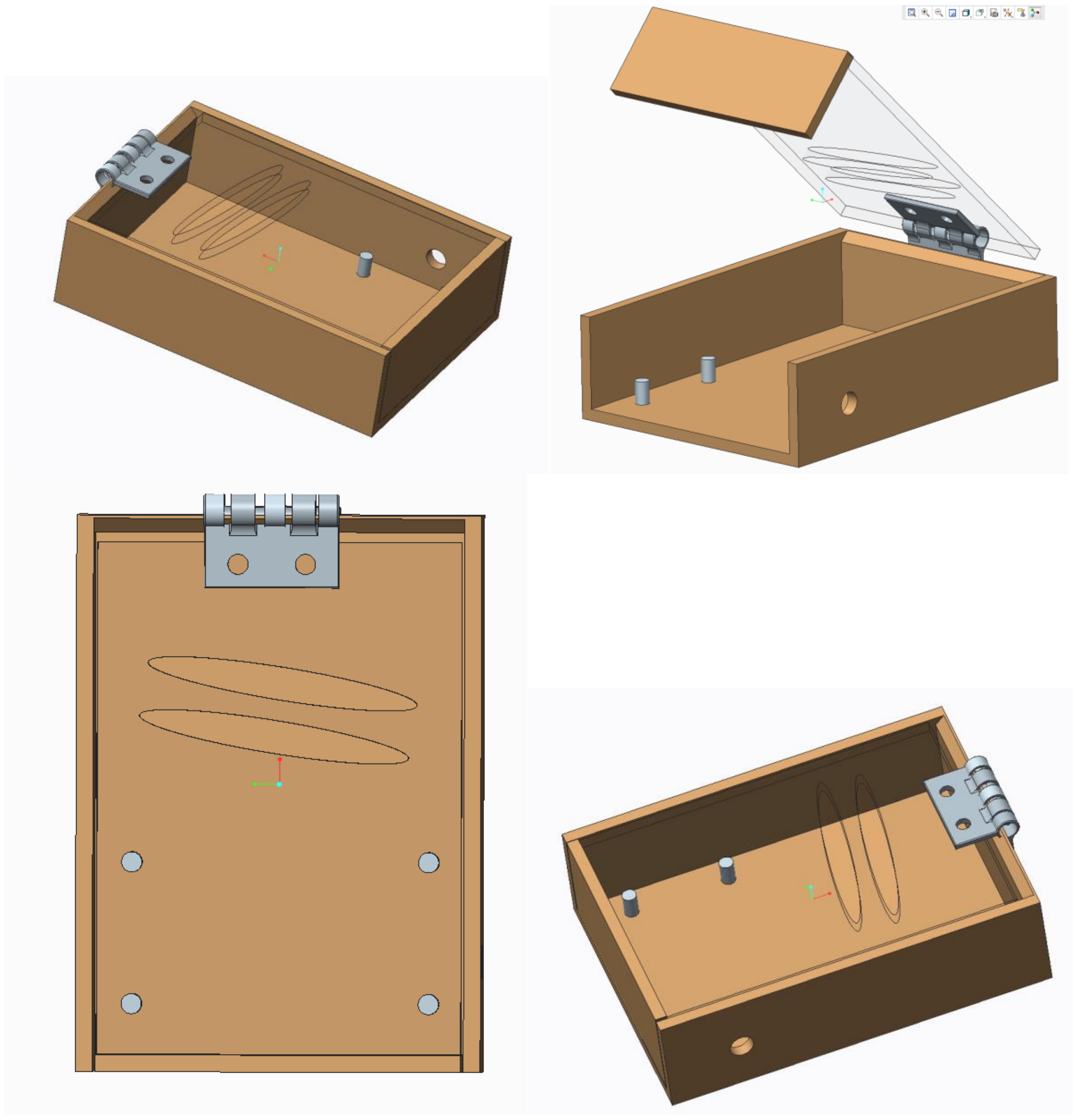


Figure 34 - Sensor Pod Enclosure Model

6.3.2. Main Control Unit

The Main Control Unit for the MHMS is the ever so popular Raspberry Pi 2 B. Because this is such a popular board, there are many cases in the market place right now. For the MHMS, a case with the ability be mounted on a wall and to internally mount the raspberry pi was the most beneficial which is why we went with the SB components Premium Clear Case for our project. This allowed access to all the ports we needed, provided ample airflow and gave us mounts to screw the pi camera so it was facing front. By using this cheap off the shelf case, our prototype will remain inexpensive yet functional.

6.4. Printed Circuit Board

The printed circuit board of this project is the interface board, which is described in section 4.3.1 Interface Board. There were several things to consider when laying out the components for this board:

- Care must be taken not to accidentally obscure or otherwise alter the properties of the RFduino antenna.
- For programming purposes, an off-the-shelf RFduino programming board will be used. However, its interfaces consists of two rows of pins. The interface board PCB must carefully match this interface to allow direct plugging of the programming board into the interface board.
- The interface board size should be as compact as is possible in order to reduce the size each sensor requires.
- The board will pass power to the sensors. In the case of the CO sensor, which requires significantly higher currents than the other sensors, the traces carrying this power must be sufficiently large.
- The board should possess appropriate power connectors to allow connection of a transformer output (wall wart, for the CO sensor's high power needs), or power from a coin cell battery for low-power sensors.

6.4.1. Fabrication Method

There were several fabrication needs for this project. One is that of the PCB manufacturer. Having a PCB is a requirement of this project, and the part of the design selected to be crafted into a PCB is the interface board.

In addition to the interface board, as described in the sensor hardware design sections the sensors require additional circuitry which do not reside on the interface board to avoid expense and including unused footprints and/or components. Put another way, buying a single copy of a PCB is quite expensive, and for each sensor we would either need to create a single custom PCB, or introduce additional size and footprints onto our interface board, making it larger in size to accommodate all parts, when only a fraction of them will ever be used at one time, for any given individual sensor. The manufacturing choices are depicted in Figure 35 - Manufacturing Configuration.

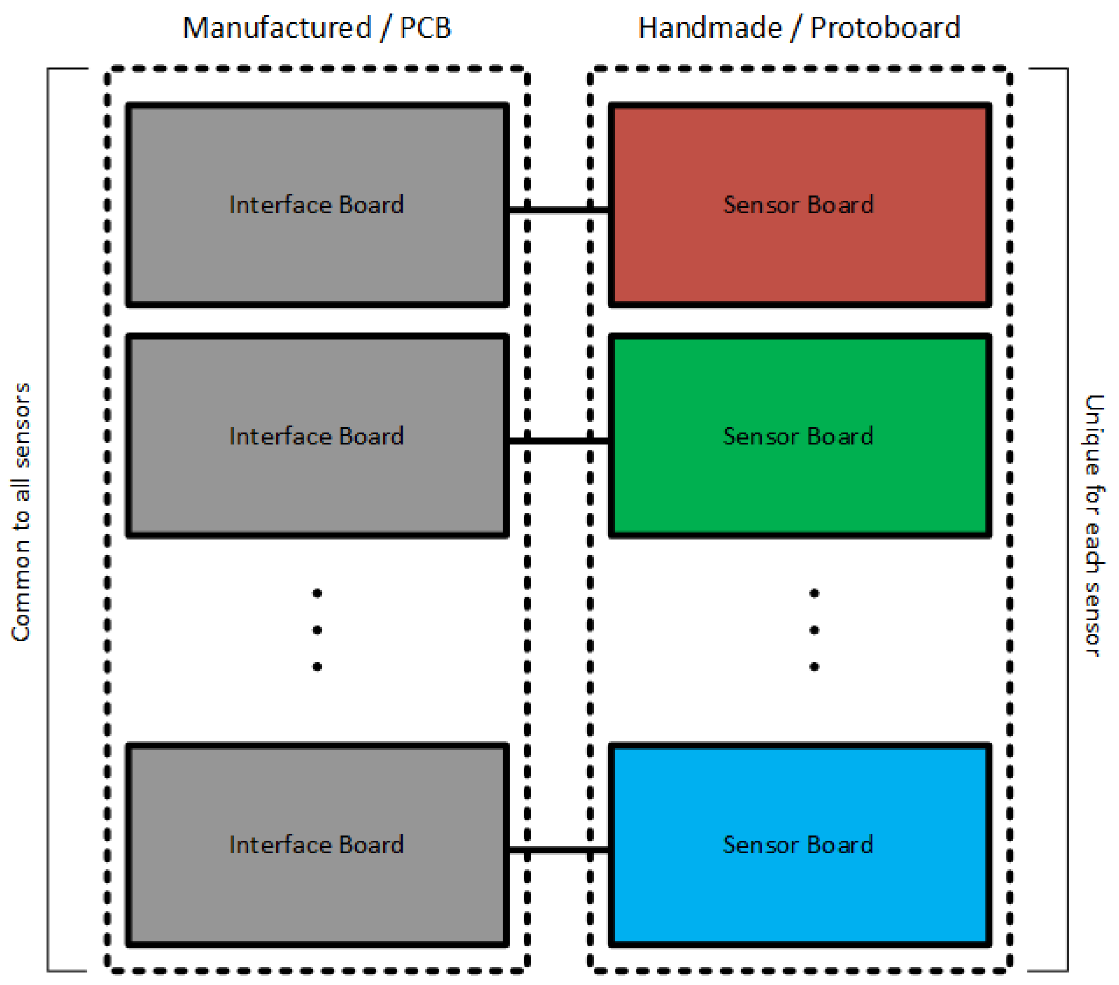


Figure 35 - Manufacturing Configuration

As such, these sensor boards were created on solderable protoboards, allowing for the running of solder or wires to connect different holes or rails in order to create the circuit. This is deemed as the best course of action because the number of "auxiliary" components required for each sensor is small, and they are not sufficiently high-speed or otherwise requiring the physical benefits associated with a PCB.

6.4.1.1. PCB Manufacturers

The following sections outline the comparison of available PCB manufacturers in order to assess which is most appropriate for this particular project. The following assumptions govern this analysis and comparison:

- Board dimensions: 2 inches by 4 inches
- Number of layers: 2
- Quantity: 5 (3 for sensors, 2 spare)

6.4.1.1.1. OSH Park

OSH Park (https://oshpark.com/) is located in the USA, and offers free shipping anywhere in the world. The shipping occurs within 12 days from ordering, for relatively quick turnaround. OSH Park's pricing and specifications are shown below:

- $5 per square inch, includes 3 copies, free shipping
    - Estimated price: $80 ($40 per 3)
- FR4 170Tg/290Td boards
- 6 mil traces with 6 mil spacing
- 13 mil drills with 7 mil annular rings
- Internal cutouts allowed and supported
- ENIG (gold) finish
- 1.6 mm thickness, 1 ounce copper both sides

6.4.1.1.2. ExpressPCB

ExpressPCB (http://www.expresspcb.com/) has a unique offering in that it provides free CAD software that allows schematic capture all the way through board layout, to accompany their manufacturing offerings. This particular aspect does not offer much benefit to this project, which is using other schematic capture and PCB layout tools.

- $61+($0.70*number of boards*board area)+($1.00*number of boards)+shipping
    - Estimated price: $100.60 + shipping
- FR4 boards
    - No silkscreen or solder mask
- 6 mil traces with 6 mil spacing
- 14 mil drills
- Internal cutouts not allowed
- Tin/lead finish
- 1.7 mm thickness, 1 ounce copper both sides

6.4.1.1.3. Seeed Studio

Seeed studio (http://www.seeedstudio.com/depot/), instead of providing specifications of the manufacturing process, provides a tool that estimates cost directly for your order. Here are the results of the estimation:

- Estimated price: $51.05 + shipping
- Unknown trace size and spacing
- 13 mil drills with 7 mil annular rings
- Tin/lead finish
- 1.6 mm thickness, 1 ounce copper both sides

6.4.1.2. PCB Manufacturer Conclusion

While the three compared PCB manufacturing companies have their own strengths and weaknesses, this project was looking for a low-volume, low-cost, high quality manufacturing experience. While these are all competing factors, OSH Park provided a very high quality offering while still keeping competitive prices, especially for low-quantity production.

As such, the project PCB (the interface board) was purchased from OSH Park.

6.4.2. Board Assembly

Assembly of the boards was be done in a garage of one of the group members. Said group member possesses the necessary soldering equipment and space required to perform such tasks.

In an effort to ease the difficulty of soldering the components onto the board, through-hole and larger pitch surface mount (for example, SO-8) parts were selected when selecting packages for circuit components. Leadless packages and otherwise difficult-to-hand-solder packages will be avoided if at all possible. Soldering was performed with a variable temperature, Hakko Fx888 soldering station with a chisel and pencil tip, using 60/40 tin lead solder and flux.

The layout of parts on the sensor boards (solderable protoboards) was captured in drawings and labelled prior to assembly of the boards, for troubleshoot, debug, and documentation purposes. Parts were connected either by the rows of the protoboard, much like in a typical breadboard, if available, properly insulated point-to-point wires, or solder bridges formed between the plated through holes of the board.

6.5. Software Implementation

The following sections describe how the software design was implemented to create a functional prototype of the Modular Home Monitoring System.

6.5.1. Sensor

In addition to the typical Arduino libraries, the RFduino has access to the RFdunoBLE library. The RFduinoBLE library made it simple to use the device through functions like RFduinoBLE.send() to send data, RFduinoBLE.txPowerLevel() to control transmit power consumption, and RFduinoBLE.begin() to start the BLE stack and start advertising. Because the RFduino is limited to 128kb of flash memory, there was a significant limitation to the size of firmware running on the device. Code had to be size efficient as well as power efficient. To increase the power efficiently we used sleep and wake functions. The RFduino library includes functions like RFduino_ULPDelay() which puts the device in an ultra-low power state for a specified amount of time and RFduino_pinWake() where the device wakes when it receives a signal on a specified GPIO pin.

Using this powerful RFduino library, the group developed software “sketches” that follows the flowcharts in section 5.1. (Sketches are Arduino programs). Because of the differences in the sensors, the RFduino’s sketches will be different depending on which sensor that Interface Board will be working with.

Sketches were then uploaded to the RFduinos using the RFduino USB programming shield connected to the headers on the Interface Board. Four signals are used to program the RFduino, thus the headers will connect the /Reset, VCC, TxD and RxD signals from the USB Shield to the corresponding lines on the IB header.

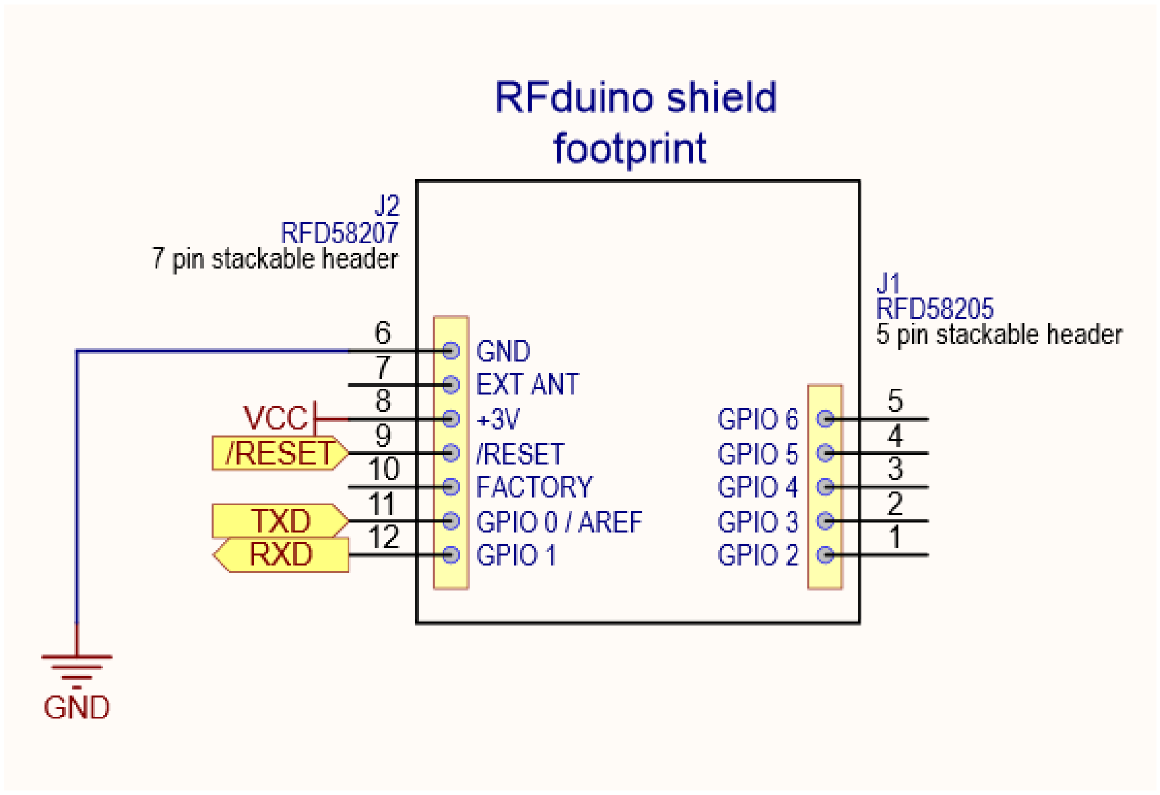


Figure 36 - RFduino USB Programming Shield Signals (Used with permission from RFduino Co.)

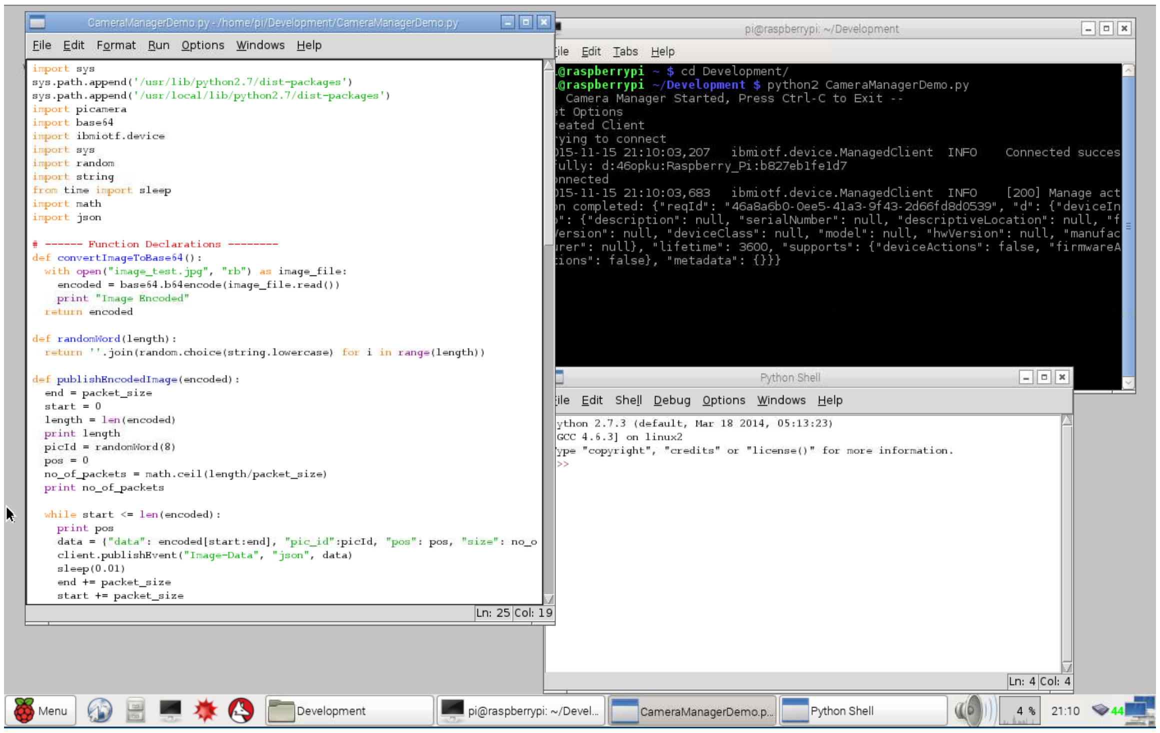
6.5.2. Main Control Unit

To build a functioning prototype of the Modular Home Monitoring System, the group had to develop proper software for the Main Control Unit. As discussed in section 5.3. The MCU is responsible for passing data from the sensor pods to the cloud software and sending an image stream from the camera to the user interface.

The communication handling software must had to have access to Bluetooth communication functions natively on the Raspberry Pi. After much research, it was decided that Python would be used to write this software. Python was recommended by many users of Raspberry Pi because it is a lightweight, multi-paradigm programming language that is well supported by Raspbian, the OS that is recommended for use on Raspberry Pi. By using Python, libraries like Python-Bluez were able to be incorporated which allow for simple interfacing to the Bluetooth Dongle connected to the Pi. In addition, a pre-made python libraries from IBM, ibmiotf allows for interfacing with the IBM Internet of Things Foundation which is how the MCU sends the sensor's data to the cloud application and how the image streaming software sends pictures to the MQTT Client on the web user interface. Using Python-Bluez ibmiotf and other supporting libraries, the MCU is able to perform its function in the system.

The camera streaming software was likewise written in python using the ibmiotf library, however it has a special feature where it only sends the image stream when the user presses the button on the web page to do so. In order to implement this functionality for the prototype, we used the client.commandcallback function which allowed us to specify a function to execute whenever a command was sent to the MCU. That function was then responsible for setting a flag that told the main thread to start or stop the image stream.

Python code development was done directly on the Raspberry Pi using the Integrated Development Environment (IDLE). IDLE was chosen because it is extremely simple and lightweight with no frills that many IDE's now a days have. Even though it is lightweight it still comes with some useful features for novice Python programmers such as an integrated debugger and auto-completion. Programming on the Raspberry Pi directly would be nearly impossible without being able to use a VNC connection which is why the group used TightVNC on a desktop computer on the same network as the Pi to connect graphically to develop.


CameraManagerDemo.py - /home/pi/Development/CameraManagerDemo.py
File Edit Format Run Options Windows Help
Ln: 25 Col: 19
pi@raspberrypi: ~/Development
ile Edit Tabs Help
Python Shell
ile Edit Shell Debug Options Windows Help
Ln: 4 Col: 4
Menu
Development
pi@raspberrypi: ~/Devel...
CameraManagerDemo.p...
Python Shell
4 %
21:10

```
import sys
sys.path.append('/usr/lib/python2.7/dist-packages')
sys.path.append('/usr/local/lib/python2.7/dist-packages')
import picamera
import base64
import ibmiotf.device
import sys
import random
import string
from time import sleep
import math
import json

# ------ Function Declarations --------
def convertImageToBase64():
  with open("image_test.jpg", "rb") as image_file:
    encoded = base64.b64encode(image_file.read())
    print "Image Encoded"
  return encoded

def randomWord(length):
  return ''.join(random.choice(string.lowercase) for i in range(length))

def publishEncodedImage(encoded):
  end = packet_size
  start = 0
  length = len(encoded)
  print length
  picId = randomWord(8)
  pos = 0
  no_of_packets = math.ceil(length/packet_size)
  print no_of_packets

  while start <= len(encoded):
    print pos
    data = {"data": encoded[start:end], "pic_id":picId, "pos": pos, "size": no_o
    client.publishEvent("Image-Data", "json", data)
    sleep(0.01)
    end += packet_size
    start += packet_size
```

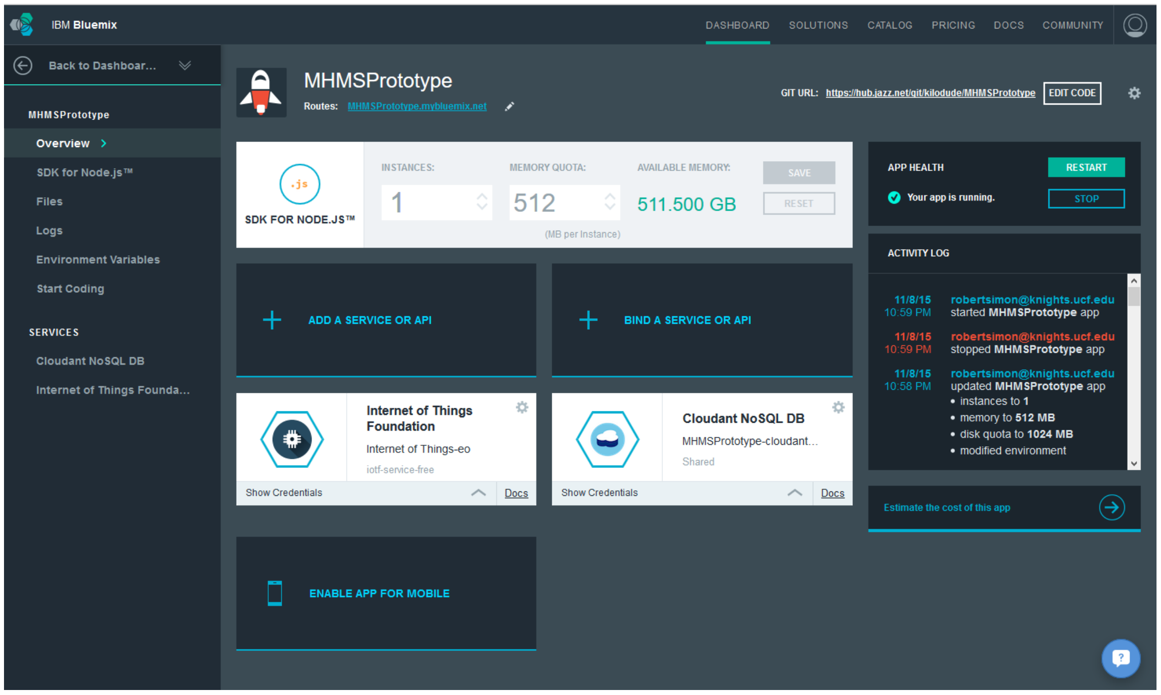
```
.@raspberrypi ~ $ cd Development/
.@raspberrypi ~/Development $ python2 CameraManagerDemo.py
 Camera Manager Started, Press Ctrl-C to Exit --
et Options
reated Client
rying to connect
015-11-15 21:10:03,207   ibmiotf.device.ManagedClient  INFO    Connected succes
fully: d:46opku:Raspberry_Pi:b827eb1fe1d7
onnected
015-11-15 21:10:03,683   ibmiotf.device.ManagedClient  INFO    [200] Manage act
on completed: {"reqId": "46a8a6b0-0ee5-41a3-9f43-2d66fd8d0539", "d": {"deviceIn
o": {"description": null, "serialNumber": null, "descriptiveLocation": null, "f
Version": null, "deviceClass": null, "model": null, "hwVersion": null, "manufac
urer": null}, "lifetime": 3600, "supports": {"deviceActions": false, "firmwareA
tions": false}, "metadata": {}}}
```

```
ython 2.7.3 (default, Mar 18 2014, 05:13:23)
GCC 4.6.3] on linux2
ype "copyright", "credits" or "license()" for more information.
>>
```

Figure 37 - IDLE running on Raspberry Pi viewed through TightVNC

6.5.3. Cloud Application

The IBM Bluemix platform provides many of "Services" to aid in the development of any project or application. We have implemented provided services into our project prototype to ensure all the requirements of the cloud application are met. First of all, the MHMS uses the IBM Internet of Things Foundation to connect the MCU to the cloud and allow simple communication using a REST API. Once we registered the MCU with the IoT foundation website, we created an application with Node-RED. Node red allows users to "wire" together hardware devices on the Internet of Things with APIs and other online services. We used node red as our main control for our cloud application. The Node-Red flow has the responsibility of decoding messages, placing data in the database, and sending alerts.



INSTANCES: 1
MEMORY QUOTA: 512
AVAILABLE MEMORY: 511.500 GB

Your app is running.

```
11/8/15    robertsimon@knights.ucf.edu
10:59 PM   started MHMSPrototype app

11/8/15    robertsimon@knights.ucf.edu
10:59 PM   stopped MHMSPrototype app

11/8/15    robertsimon@knights.ucf.edu
10:58 PM   updated MHMSPrototype app
           • instances to 1
           • memory to 512 MB
           • disk quota to 1024 MB
           • modified environment
```

Figure 38 – The MHMS App Console with IoTF and Cloudant DB Service

Since none of the MHMS team members were experienced in database management, it was decided early on that the cloud application would use the Cloudant NoSQL data base service to provide easy to use access to a JSON data layer in which alerts/status messages can be saved and user data can be stored for login authentication. Cloudant NoSQL DB is a NoSQL database as a service (DBaaS). It is built to handle a wide variety of data types like JSON, full text and geospatial. It is advertised to be an operational data store optimized to handle concurrent reads and writes, and provide high availability of data durability.

Finally, the Twilio third party service was integrated into the application to allow for SMS messages to be sent to the user when a hazardous situation is detected. Twilio SMS allows users to programmatically send, receive, and track messages worldwide. Once a user (or organization in this case) makes an account, a phone number is assigned that can be used in API calls within all types of software. The Twilio REST API is served over HTTPS with a base URL of https://api.twilio.com/2010-04-01. Sending a SMS text message is made very simple with the use of the Twilio Node-Red node which took care of the API specifics for us. Because we do not want users being bombarded with text messages, a check is done to detect if this particular alert has been sent to the user already within the past 10 minutes.

6.5.4. User Interface

The prototype for the user interface is accessible and hosted through IBM Bluemix. Bluemix applications have a HTML front end and come with a standard URL of myappname.mybluemix.net. This was sufficient for building a prototype of the Modular Home Monitoring System as it is well integrated into the other Bluemix services that we used and required no extra work in setting up like an external web host would.

The user interface is written in HTML5/ JavaScript for the front end, and the back end will be written in PHP. The styling on the UI is done using Cascading Style Sheets (CSS). Because the user will be checking this software often, we tried to give the website a modern look to it. However, because this user interface prototype is mainly focused on functionality, the priority in development for the UI was to get a basic, functional version done first, then add any styling later.

To view the image stream from the MCU on the webpage, a PAHO MQTT client was embedded into the website through JavaScript. The client sends commands to the MCU to start or stop the video stream, receives the image files as base 64 data chunks and recompiles it to show to the user. An IBM Recipe was followed to create this functionality and pass image data through IBM IoTF. [40]

Development of the website was done using notepad ++ for the HTML, CSS, JavaScript and PHP code. Files were then deployed to Bluemix using the cloud foundry command line interface which was downloaded directly from the Bluemix console. Github was also used to manage different versions of the build and keep track of changes.

7. Testing

When designing a distributed system like the MHMS, comprehensive testing is necessary in order to meet exacting standards for system uptime and sensor accuracy. Fortunately, the modularity of the MHMS building block works to our advantage, allowing us to test each piece in isolation before complete integration, which provides stronger guarantees about the robustness of the final product and greatly simplifies debugging. Consequently, we have designed a test procedure that exercises all important capabilities of the system, and split it into sections for each relevant part.

7.1. Sensor Functional

The most basic parts of the MHMS are its sensors, which provide the raw data that the main controller consumes, whether by logging, generating alerts, or taking other action. For the purposes of testing, the sensors will be taken out of the system and configured to output data to the console. Console output will be logged by an attached PC. Unfortunately, we did not have the time or the budget to exhaustively test every sensor. In particular, the carbon monoxide sensor would require an airtight chamber (with a gas cylinder containing CO at a known concentration) and specialized equipment in order to verify the accuracy of its measurements. For this reason, we have instead devised a test procedure which will produce a binary YES/NO response to the presence of CO gas, even though the sensor is capable of giving a quantitative measure of CO concentration.

<u>Sensor test: Humidity</u>

Module under test: Humidity sensor

Equipment: Commercial relative humidity sensor, commercial humidifier

Procedure:

1. The sensor module under test is placed in a closed container (not necessarily airtight) with the humidifier and the commercial sensor. Care should be taken to place the two sensors at approximately equal distance from the humidifier, to avoid bias in the results.
2. Baseline readings from both sensors are taken.
3. The humidifier is activated, and readings from each sensor are taken at 20 second intervals.
4. After 2 minutes, the humidifier is deactivated, and readings from each sensor are again taken at 20 second intervals, for another 2 minutes.

Expected Results: Our sensor should agree with the commercial one up to a relative humidity difference of 5 % (in either direction). Assuming a sufficiently accurate commercial sensor, this procedure should suffice for proving the responsiveness and correctness of our humidity sensor. Record all results on the test sheet shown in Table 13 - Humidity Sensor Test Sheet.

Table 13 - Humidity Sensor Test Sheet

Test Number	**Unit ID**	**RH Value (%) – MHMS Sensor**	**RH Value (%) – Commercial Sensor**	**Notes**
Baseline reading				
1				
2				
3				
4				

5				
6				
7				
8				
9				
10				
11				
12				

<u>Sensor test: Smoke</u>

Module under test: Smoke sensor

Equipment: Wizard Stick (hand held fog machine)

Procedure:

1. The sensor module under test is mounted on a vertical surface (such as a wall).
2. The wizard stick is activated, and held approximately 10 cm below the sensor, and extinguished. The fog should simulate an early-stage smoldering fire.
3. If the sensor triggers within 5 seconds, the test is marked as a "PASS", otherwise, the test is marked "FAIL". The test will be repeated 10 times, and the results for are recorded. At least 20 seconds should pass between each test, in order to allow the fog to dissipate.

Expected Results: At least 9 out of the 10 iterations of the procedure should have triggered the sensor. If the sensor ever activates during the interval between iterations, then the entire sensor test should be considered a failure (avoiding spurious alarms is an important objective for any fire protection system). Record all results on the test sheet shown in Table 14 - Smoke Sensor Test Sheet.

Table 14 - Smoke Sensor Test Sheet

Test Number	**Unit ID**	**Pass (P)/Fail (F)**	**Notes**
1			
2			

3			
4			
5			
6			
7			
8			
9			
10			

<u>Sensor test: Carbon Monoxide</u>

Module under test: Carbon monoxide sensor

Equipment: Automobile

Procedure:

1. An outdoor location, far from roads and motor vehicles (or any other potential sources of combustion by-products) should be chosen for the test. The air should be relatively still, as a strong breeze could clear the CO before the sensor has a chance to respond.
2. A baseline reading is taken. The sensor should indicate no more the 10 ppm, since it is outdoors and far from any CO sources. This reading represents the atmospheric CO concentration.
3. A car is turned on, and its exhaust brought approximately 10 cm below the sensor. The incomplete combustion will produce an elevated CO level for the sensor to detect.
4. If the sensor triggers within 5 seconds, the test is marked as a “PASS”, otherwise, the test is marked “FAIL”. The test will be repeated 10 times, and the results for are recorded. At least 20 seconds should pass between each test, in order to allow the CO concentration to return to normal.

Expected Results: The baseline reading should have yielded a result of no greater than 10 ppm. At least 9 out of the 10 iterations of the above procedure should have triggered the sensor. If the sensor ever activates during the interval between iterations, then the entire test should be considered a failure. Record all results on the test sheet shown in Table 15 – CO Sensor Test Sheet.

Table 15 – CO Sensor Test Sheet

Test Number	**Unit ID**	**Pass (P)/Fail (F)**	**Notes**
Baseline			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

7.2. Sensor to Main Control Unit Communication

Second in importance only to the collection of data is communication with the MCU. While gathering data directly from the console is acceptable in early testing, wireless communication of sensor data to the MCU is crucial to our project. Data transfer must be robust, correct, and fast enough to collect data from all sensors operating simultaneously.

<u>Communication test</u>

Procedure:

1. A location must be selected for the test, and the full complement of sensors must be deployed nearby. Any typical structure should be suitable, but each sensor should be within 20 meters of the base station.
2. All sensors are powered on and paired with the MCU.
3. Communication with the MCU is verified by logging in via SSH or VNC (since the web interface is not the subject of this test) and checking the relevant log files. Each sensor should both be listed as active by the base station, and producing readings at various intervals. The exact content and timing of the readings is not important, and will vary between sensor types.
4. At the end of 1 hour, the sensors should still be connected and producing readings.

Expected Results: After one hour has elapsed, the sensors should still be connected to the base station and sending periodic updates. If this condition is met, the communication infrastructures should reliable enough to ensure that our sensor data reaches the MCU.

7.3. Camera Functional

Though the camera will share a housing with the MCU, and is consequently unlikely to experience as many communication interruptions as our wireless sensors will, testing is still necessary to verify functionality.

Camera test

Procedure:

1. The web interface of the MCU should be accessed by entering the URL "mhmsPrototype.mybluemix.net" into a standard web browser. Both desktop and mobile platforms should be used for this test.
2. The web page should be checked to ensure that the camera is producing a live image. In order to verify that the image is live and the framerate is within acceptable margins, a hand should be waved in front of the camera. If the image fails to load or does not update, the test is a failure.

Expected Results: Both mobile and desktop browsers should load a page with a periodically updating image from the camera.

7.4. System Functional

After all of the system components have been individually tested and found to operate satisfactorily in isolation, it is necessary to verify the performance of the complete system.

Procedure:

1. The sensors should be deployed and paired as in the sensor test procedure.
2. The web interface of the MCU should be accessed by entering the URL "mhmsPrototype.mybluemix.net" into a standard web browser (where 192.168.0.101 is the IP address of the MCU). Both desktop and mobile platforms should be used for this test.
3. The camera should be tested as in the camera test procedure.
4. Each sensor should show as "ACTIVE" in the monitoring window. Additionally, periodic status updates should be visible in the alerts pane.
5. Steps 2 through 4 should be repeated after 1 hour has elapsed.

Expected Results: The web interface should accurately represent the status of all sensors, including the camera. The sensors should not lose connectivity or cease producing readings over the duration of the test.

8. Related Standards

The following related standards have been compiled through research into various standardization organizations and have direct application to the Modular Home Monitoring System.

- ANSI C84.1
  - Electric Power Systems and Equipment – Voltage Ratings (60 Hertz)
- Electronic Code of Federal Regulations Title 47 - Telecommunication
  - Part 15 - Radio Frequency Devices
- United States Code Title 15
  - Starting at section 1261 – FHSA requirements
- Bluetooth 4.0
  - Bluetooth Standard that includes Bluetooth Low Energy protocol.
- ISO 7240-15:2014 Fire detection and alarm systems – Part 15: Point-type fire detectors using smoke and heat sensors
  - Specifies requirements, test methods, and performance criteria for point-type fire detectors using smoke and heat sensors, incorporating in one mechanical enclosure at least one smoke sensor and at least one other sensor which responds to heat, and in which the signal(s) of the smoke sensor(s) is (are) combined with the signal(s) of the heat sensor(s).
- IEEE 802.11n Wifi
  - Communications on the 2.4 and 5 GHz range from 54 to 600 Mbps
- IEC 61188-5-1 PCB Assemblies – Design and Use
  - Provides information on land pattern geometries used for the surface attachment of electronic components.
- IEC 61191-1 Printed Board Assemblies Part 1 General Specifications
  - Requirements for soldered electrical and electronic assemblies using surface mount and related assembly technologies
- IEC 60065 Ed. 8.0 b:2014 Audio, video and similar electronic apparatus – Safety Requirements
  - This applies to electronic apparatus designed to be fed from the mains, from a supply apparatus, from batteries or from remote power feeding and intended for reception, generation, recording or reproduction of audio, video and associated signals. It also applies to apparatus designed to be used exclusively in combination with the above-mentioned apparatus. This standard primarily concerns apparatus intended for household and similar general use but which may also be used in places of public assembly such as schools, theatres, places of worship and the workplace professional apparatus intended for use as described above is also covered unless falling specifically within the scope of other standards.
- CSI-3 Camera Serial Interface Standard
  - Specifies serial communication between camera subsystems and systems like mobile phones and other devices.
- IEC 62676-2-2 Ed. 1.0 b:2013 Video surveillance systems for use in security applications

    - Specifies a compliant IP video protocol based on HTTP and REST services. Video transmission devices are often equipped with web servers that respond to HTTP requests. The HTTP response may contain XML content (for GET actions), XML response information (for SET actions), or various text/binary content (for retrieval of configuration data, etc.). REST is an approach to creating services that expose all information as resources in a uniform way. A video transmission device supporting compliance to the requirements of this standard based on HTTP and REST Services as described in this document is declared as compatible to nIEC 62676-2 HTTP and REST interoperability.
- BSR/IEEE 2700-201x Standard for Sensor Performance Parameter Definitions
    - This document provides a common framework for sensor performance specification terminology, units, conditions and limits. The specific sensors discussed in this document are Accelerometer, Magnetometer, Gyrometer/Gyroscope, Barometer/Pressure Sensors, Hygrometer/Humidity Sensors, Temperature Sensors, Ambient Light Sensors, and Proximity Sensors.
- BS IEC 60287-3-2:2012 Electric cables
    - Calculation of the current rating. Sections on operating conditions. Economic optimization of power cable size (British Standard)
- ISO 10014:2006 Quality management
    - Guidelines for realizing financial and economic benefits
- IEC/TR 62258-3 Ed. 2.0 b:2010 Semiconductor die products
    - Recommendations for good practice in handling, packing and storage
- IEC 60086-1 Ed. 11.0 b:2011 Primary batteries - Part 1: General
    - IEC 60086-1:2011 is intended to standardize primary batteries with respect to dimensions, nomenclature, terminal configurations, markings, test methods, typical performance, safety and environmental aspects. The object of IEC 60086-1 is to benefit primary battery users, device designers and battery manufacturers by ensuring that batteries from different manufacturers are interchangeable according to standard form, fit and function.
- IEC 60086-4 Ed. 4.0 b:2014 Safety of lithium batteries
    - IEC 60086-4:2014 specifies tests and requirements for primary lithium batteries to ensure their safe operation under intended use and reasonably foreseeable misuse.
- IEC 60747-14-1 Ed. 1.0 Semiconductor devices - Part 14-1
    - Semiconductor sensors - General and classification
- IEC 61207-1 Ed. 2.0 b:2010 Expression of performance of gas analyzers
    - Specifies the terminology, definitions, and requirements for statements by manufacturers and tests that are common to all gas analyzers. It is applicable to analyzers specified for permanent installation in any location (indoors or outdoors) and to such analyzers utilizing either a sample handling system or an in situ measurement technique.
- ISO 7240-27:2009 Fire detection and alarm systems

    - Point-type fire detectors using a scattered-light, transmitted-light or ionization smoke sensor, an electrochemical-cell carbon-monoxide sensor and a heat sensor
- ISO 20282-1:2006 Ease of operation of everyday products
    - Provides requirements and recommendations for the design of easy-to-operate everyday products, where ease of operation addresses a subset of the concept of usability concerned with the user interface by taking account of the relevant user characteristics and the context of use.
- BSR/IEEE 2413-201x Standard for an Architectural Framework for the Internet of Things (IoT)
    - This standard defines an architectural framework for the Internet of Things (IoT), including descriptions of various IoT domains, definitions of IoT domain abstractions, and identification of commonalities between different IoT domains. The architectural framework for IoT provides a reference model that defines relationships among various IoT verticals (e.g., transportation, healthcare, etc.) and common architecture elements.
- ISO/IEC 13249-1:2007 Database languages - SQL multimedia and application packages - Part 1: Framework
    - Defines a number of packages of generic data types common to various kinds of data used in multimedia and application areas, to enable that data to be stored and manipulated in an SQL database. The package in each subject area is defined as a part of ISO/IEC 13249. ISO/IEC 13249-1:2007 defines those concepts, notations and conventions that are common to two or more other parts of ISO/IEC 13249. In particular, it describes the way ISO/IEC 9075 is used in other parts of ISO/IEC 13249 to define the user-defined types and their behavior appropriate to each subject area.
- ISO/IEC 18021:2002 User Interfaces
    - User interfaces for mobile tools for management of database communications in a client-server model
- IEC 61340-5-3 Ed. 2.0 b:2015 Protection of electronic devices from electrostatic phenomena
    - Defines the ESD protective packaging properties needed to protect electrostatic discharge sensitive devices (ESDS) through all phases of production, rework/maintenance, transport and storage. Test methods are referenced to evaluate packaging and packaging materials for these product and material properties. Performance limits are provided. This standard does not address protection from electromagnetic interference (EMI), electromagnetic pulsing (EMP) or protection of volatile materials.
- BSR/IPC 2615-200x Printed Board Dimensions and Tolerances
    - Covers dimensioning and tolerancing of electronic packaging as it relates to printed boards and the assembly of printed boards. The concepts defined in this standard are derived from American National Standard for Dimensioning and Tolerancing, ANSI/ASME Y14.5-1994 (R1999).
- BSR C63.2-201x Standard for Specifications for Electromagnetic Noise and Field
    - Calls out various specifications that are outdated and no commercially available EMI receiver will meet today. In addition, many specifications are

outdated or no longer relevant. Furthermore, harmonization with the globally used EMI test instrumentation standard CISPR 16-1-1 is desirable to ensure that a unique set of instrumentation specifications are available that can be met by a single piece of test equipment.

- BSR/IEEE 1688-200x Standard for Module Electromagnetic Interference (EMI) Testing
  - The scope of the standard is limited to EMI levels and test methods for replaceable electronic modules (i.e. cards, electronic subunits).

9. Administrative

9.1. Milestones

The milestones that lay ahead for the project consist of the following broad goals:

- Construction of project
- Testing of project
- Final presentation of project

These goals may be further broken down into sub goals, and are shown in Figure 39 - Senior Design II Goal Breakdown.

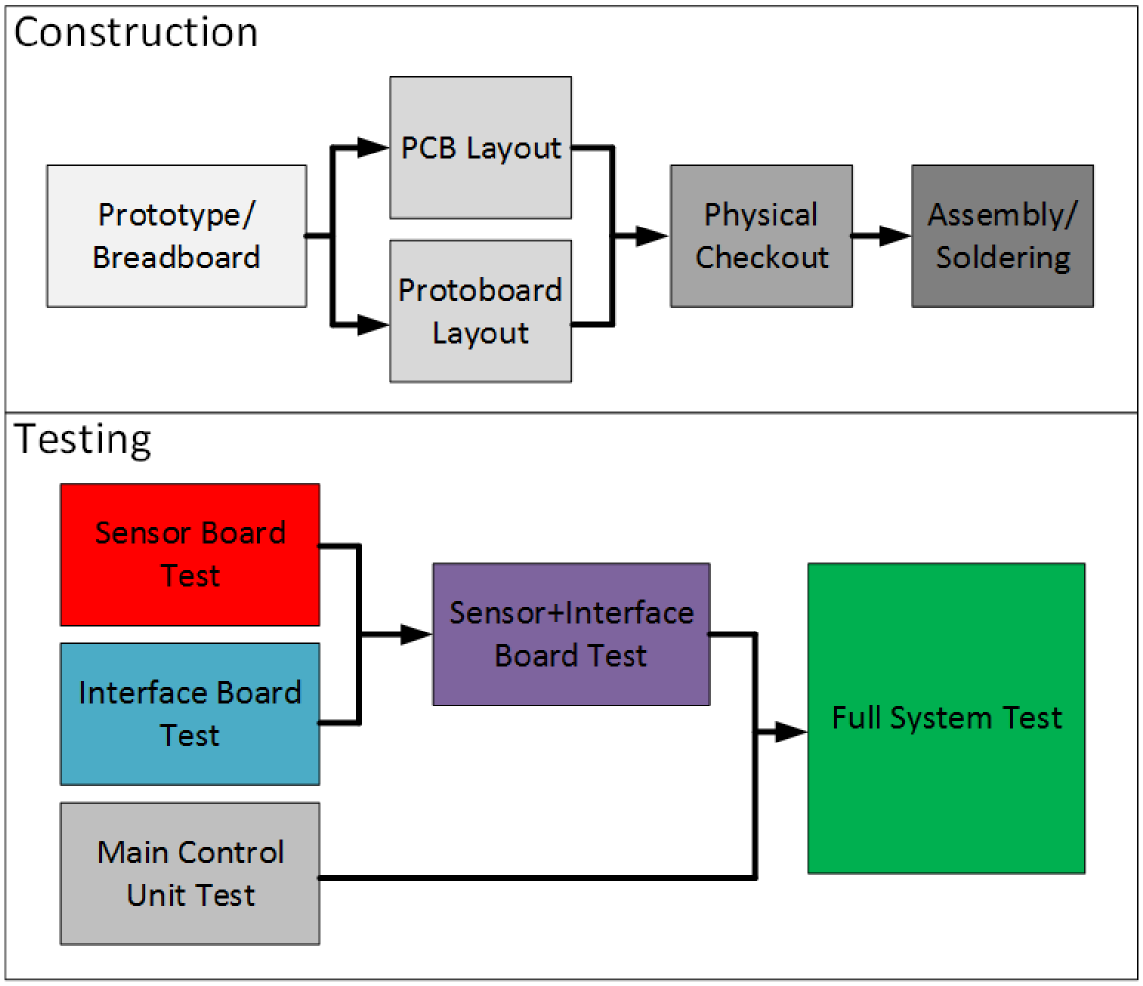
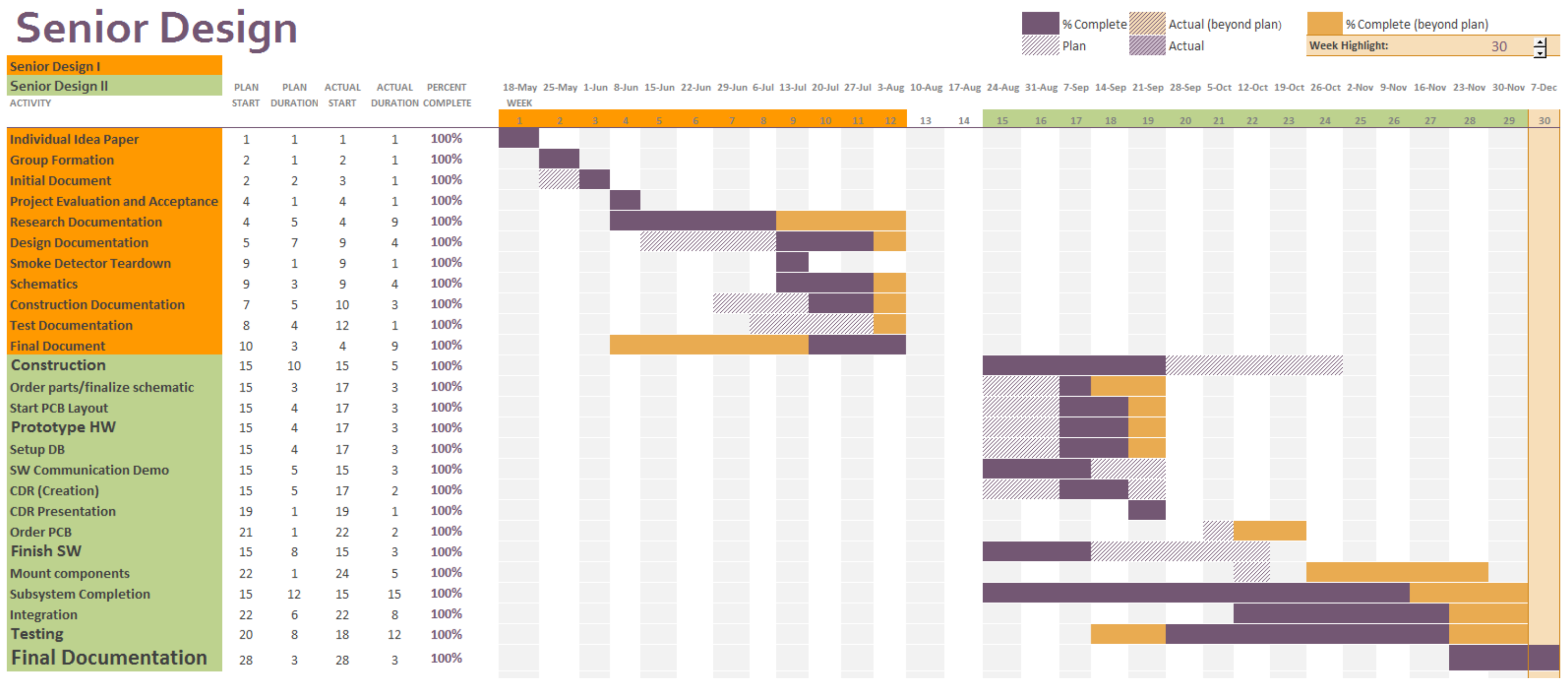


Figure 39 - Senior Design II Goal Breakdown

Furthermore, the overall schedule of both Senior Design I and II is shown in Figure 40 - Senior Design Schedule. The last week of the Senior Design I semester is the currently highlighted week in the image.



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Individual Idea Paper	1	1	1	1	100%
Group Formation	2	1	2	1	100%
Initial Document	2	2	3	1	100%
Project Evaluation and Acceptance	4	1	4	1	100%
Research Documentation	4	5	4	9	100%
Design Documentation	5	7	9	4	100%
Smoke Detector Teardown	9	1	9	1	100%
Schematics	9	3	9	4	100%
Construction Documentation	7	5	10	3	100%
Test Documentation	8	4	12	1	100%
Final Document	10	3	4	9	100%
Construction	15	10	15	5	100%
Order parts/finalize schematic	15	3	17	3	100%
Start PCB Layout	15	4	17	3	100%
Prototype HW	15	4	17	3	100%
Setup DB	15	4	17	3	100%
SW Communication Demo	15	5	15	3	100%
CDR (Creation)	15	5	17	2	100%
CDR Presentation	19	1	19	1	100%
Order PCB	21	1	22	2	100%
Finish SW	15	8	15	3	100%
Mount components	22	1	24	5	100%
Subsystem Completion	15	12	15	15	100%
Integration	22	6	22	8	100%
Testing	20	8	18	12	100%
Final Documentation	28	3	28	3	100%

Figure 40 - Senior Design Schedule

9.2. Budget and Finance

The project will be financed equally by all members, meaning that the total cost of the project will be split into equal thirds for each member to pay. The exception to this rule is if a particular member has a personal interest in keeping a certain component at the conclusion of the project. If this is the case, that member may pay for that component or components alone, and retain ownership of said component or components at the conclusion of the project. The established budget for each section of the project is shown in Table 16 - Project Budget.

Table 16 - Project Budget

Item	Cost
Main Control Unit (W/peripherals)	$112.00
3 RFduino SMT Chips	$45.00
RFduino USB Programmer	$26.00
Sensor Pod Housings	$30.00
3 Interface Board PCB Printing	$100.00
Protoboards	$30.00
Sensors	$50.00
Programs (Open source or free to use)	$0.00
Miscellaneous	$60.00
Total	$453.00

10. Appendices

A. Large Figures

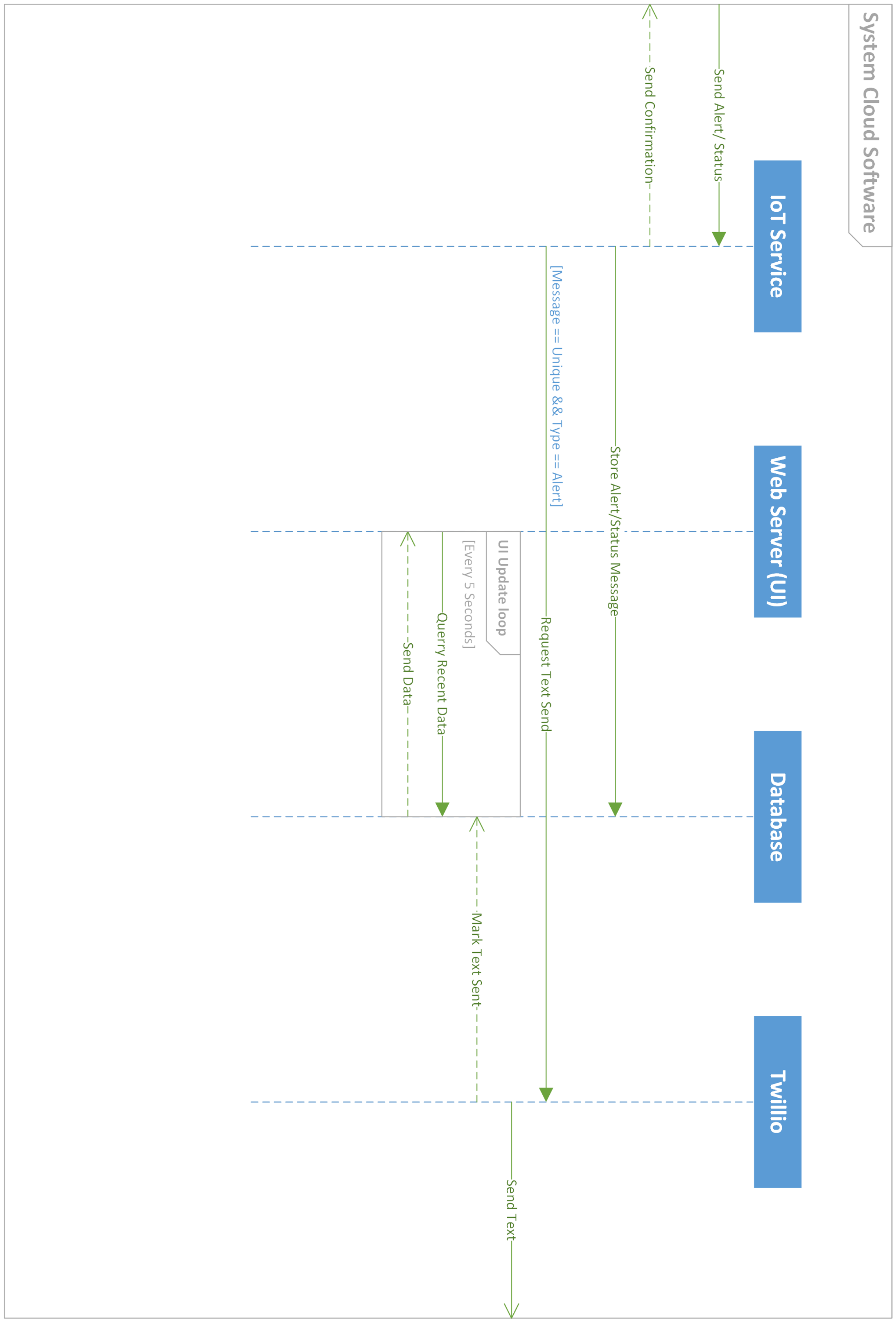
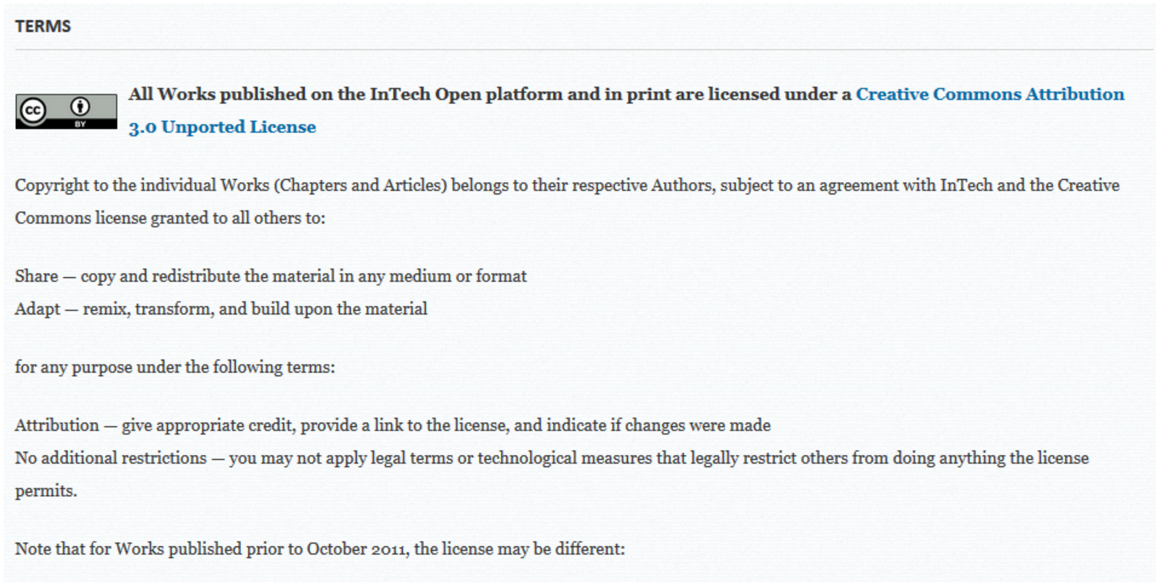


Figure 41 - Software Sequence Diagram

B. Permissions and Works Cited

The permissions appendix contains all permissions for reproduction of copyright material within this document.

Figure 3 - Example ZigBee Network, reprinted with permission from



TERMS

All Works published on the InTech Open platform and in print are licensed under a Creative Commons Attribution 3.0 Unported License

Copyright to the individual Works (Chapters and Articles) belongs to their respective Authors, subject to an agreement with InTech and the Creative Commons license granted to all others to:

Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the material

for any purpose under the following terms:

Attribution — give appropriate credit, provide a link to the license, and indicate if changes were made
No additional restrictions — you may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Note that for Works published prior to October 2011, the license may be different:

http://www.intechopen.com/books/wireless-sensor-networks-technology-and-applications/wireless-sensor-networks-to-improve-road-monitoring

Figure 4 - TCP vs UDP Header

Educational and Non-Profit Use of Copyrighted Material: If you use Microchip copyrighted material solely for educational (non-profit) purposes falling under the "fair use" exception of the U.S. Copyright Act of 1976 then you do not need Microchip's written permission. For example, Microchip's permission is not required when using copyrighted material in: (1) an academic report, thesis, or dissertation; (2) classroom handouts or textbook; or (3) a presentation or article that is solely educational in nature (e.g., technical article published in a magazine). Please note that offering Microchip copyrighted material at a trade show or industry conference for the purpose of promoting product sales does require Microchip's permission.

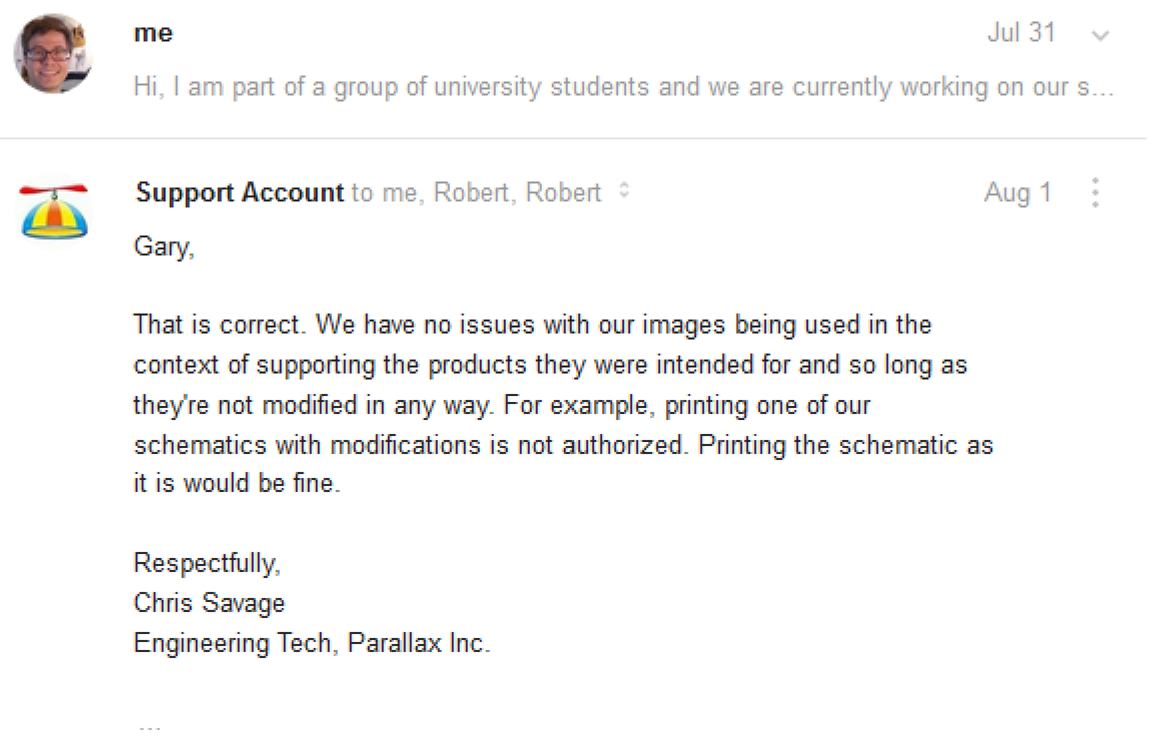
https://microchip.wikidot.com/tcpip:tcp-vs-udp

Figure 5 - User Interface Details Used with permission from ScotXW

This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

https://commons.wikimedia.org/wiki/File:Linux_kernel_INPUT_OUPUT_evdev_gem_USB_framebuffer.svg#

Figure 14 - HS1100/1 Driver and Measurement Circuit. Used with permission from Parallax.



me Jul 31

Hi, I am part of a group of university students and we are currently working on our s...

Support Account to me, Robert, Robert Aug 1

Gary,

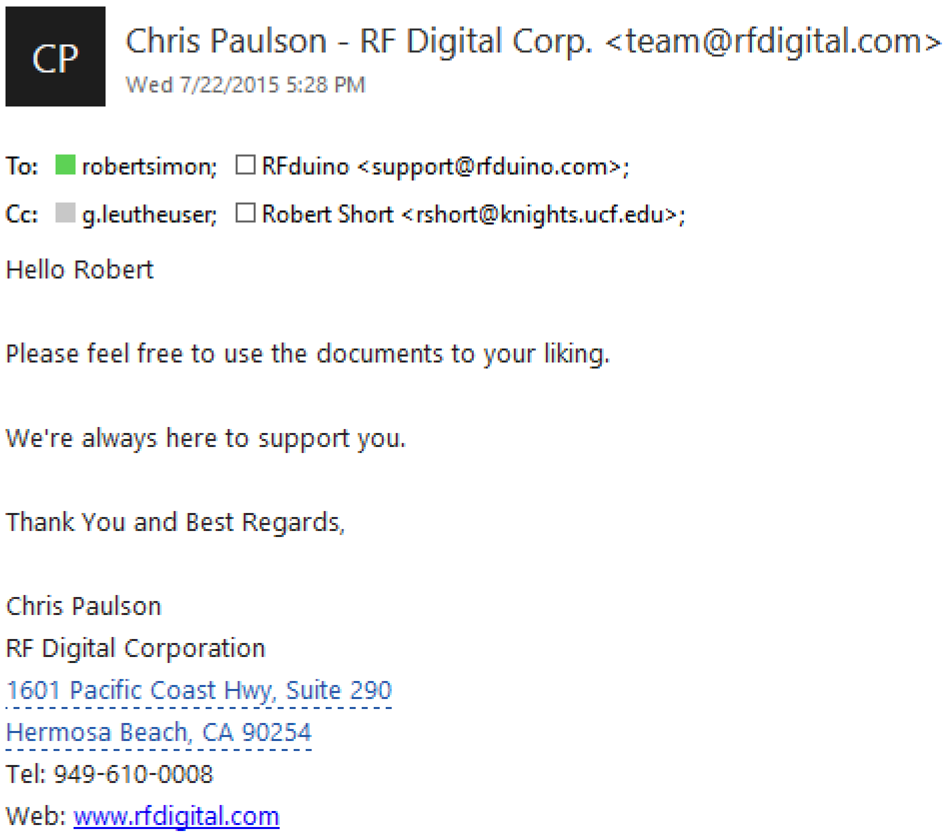
That is correct. We have no issues with our images being used in the context of supporting the products they were intended for and so long as they're not modified in any way. For example, printing one of our schematics with modifications is not authorized. Printing the schematic as it is would be fine.

Respectfully,
Chris Savage
Engineering Tech, Parallax Inc.

...

http://media.digikey.com/pdf/Data%20Sheets/Humirel%20PDFs/HS1100%20HS1101.pdf

Figure 33 - RFduino Keep out area example (Used with permission from the RFduino Co



CP Chris Paulson - RF Digital Corp. <team@rfdigital.com>
Wed 7/22/2015 5:28 PM

To: 🟩 robertsimon; ☐ RFduino <support@rfduino.com>;
Cc: ▨ g.leutheuser; ☐ Robert Short <rshort@knights.ucf.edu>;

Hello Robert

Please feel free to use the documents to your liking.

We're always here to support you.

Thank You and Best Regards,

Chris Paulson
RF Digital Corporation
1601 Pacific Coast Hwy, Suite 290
Hermosa Beach, CA 90254
Tel: 949-610-0008
Web: www.rfdigital.com

8

http://www.rfduino.com/wp-content/uploads/2014/03/RFD22301.Data_.Sheet_.11.24.13_11.38PM.pdf

Works Cited

[1] L. M. F. X. T. Y. J. Q. M. S. Ming Xu, "Implementation of a Wireless Sensor Network for Smart Homes," IEEE, Hangzhou.

[2] A. K. A. K. Nilesh Khude, "Time and Energy Complexity of Distributed Computation in Wireless Sensor Networks," IEEE, 2005.

[3] G. K. Gorkem Ozvural, "Advanced Approaches for Wireless Sensor Network Applications and Cloud Analytics," IEEE, Singapore, 2015.

[4] O. P. P. M. S. G. Mario Ribeiro, "Architecture for Modular Smart Sensor Systems," in *Sixth International Conference on Sensing Technology (ICST)*, 2012.

[5] Microsoft, "The OSI Model's Seven Layers Defined and Functions Explained," 13 June 2014. [Online]. Available: https://support.microsoft.com/en-us/kb/103884. [Accessed 7 July 2015].

[6] A. Samanta, "Communication Protocol," 23 May 2008. [Online]. Available: http://www.slideshare.net/sankhadeep/communication-protocol-arindam-samanta. [Accessed 6 July 2015].

[7] "Bluetooth Fast Facts," Bluetooth SIG, 2015. [Online]. Available: http://www.bluetooth.com/Pages/Fast-Facts.aspx. [Accessed 27 6 2015].

[8] JIMBO, "Bluetooth Basics," Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/bluetooth-basics/common-versions. [Accessed 27 June 2015].

[9] S. H. S. T. J. S. Artem Dementyev, "Power Consumption Analysis of Bluetooth Low Energy, ZigBee," Microsoft Research, Cambridge.

[10] Bluetooth SIG, "Specification of the Bluetooth System," 2 December 2014. [Online]. Available: https://www.bluetooth.org/en-us/specification/adopted-specifications. [Accessed 27 June 2015].

[11] Intel, "Helping Define IEEE 802.11 and other Wireless LAN Standards," Intel.

[12] Texas Instruments, "CC3200 Current Measurements," Texas Instruments, 7 May 2015. [Online]. Available: http://processors.wiki.ti.com/index.php/CC3200_Current_Measurements#Expected_results. [Accessed 27 June 2015].

[13] Z-Wave, "About Z-Wave," 2015. [Online]. Available: http://z-wave.sigmadesigns.com/about_z-wave. [Accessed 28 June 2015].

[14] P. Kinney, "ZigBee Technology: Wireless Control that Simply Works," 2 October 2003. [Online]. Available: http://www.mouser.com/pdfdocs/ZigBeeTechnology.pdf. [Accessed 29 June 2015].

[15] Green Peak, "ZigBee compared with Bluetooth Low Energy," [Online]. Available: http://www.greenpeak.com/company/Opinions/CeesLinksColumn19.pdf. [Accessed 29 June 2015].

[16] Texas Instruments, "A USB-Enabled System on a chip solution for 2.4-GHz IEEE 802.15.4 and ZigBee applications," June 2010. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2531.pdf. [Accessed 29 June 2015].

[17] I. Poole, "GPRS General Packet Radio Service Tutorial," Radio Electronics, [Online]. Available: http://www.radio-electronics.com/info/cellulartelecomms/gprs/gprs_tutorial.php. [Accessed 29 June 2015].

[18] ITead Studio, "GPRS Module," 6 December 2011. [Online]. Available: ftp://imall.iteadstudio.com/Modules/IM120525010_SIM900_module/DS_IM120525010_GPRS_Module.pdf. [Accessed 29 June 2015].

[19] O. K. J. suhonen, "Sensor Information Data Format (SIDF)," Tampere University of Technology, 14 September 2012. [Online]. Available: http://www.tkt.cs.tut.fi/research/gwg/openapi/sidf.html. [Accessed 6 July 2015].

[20] w3schools, "Web TCP/IP," w3schools.com, [Online]. Available: http://www.w3schools.com/website/web_tcpip.asp. [Accessed 7 July 2015].

[21] Mircorosft, "How TCP/IP Works," 28 March 2003. [Online]. Available: https://technet.microsoft.com/en-us/library/cc786128%28v=ws.10%29.aspx. [Accessed 7 July 2015].

[22] J. Postel, "User Datagram Protocol RFC 768," IETF, 1980.

[23] Beagle Board, "BeagleBone Black," [Online]. Available: http://beagleboard.org/BLACK. [Accessed 8 July 2015].

[24] Raspberry Pi, "Raspberry Pi 2 Model B," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/. [Accessed 7 July 2015].

[25] "CARBON MONOXIDE ALARM CONFORMANCE TESTING TO UL 2034: STANDARD FOR SAFETY FOR CARBON MONOXIDE ALARMS," September 2013. [Online]. Available: http://www.cpsc.gov//Global/Research-and-Statistics/Technical-Reports/Home/Carbon-Monoxide/COAlarmConformanceReportFY%202013ClearedTechReport.pdf. [Accessed 4 August 2015].

[26] L. A. Gundel, M. G. Apte and A. R. Nematollahi, "Carbon Monoxide Technology Comparison: Respone to Various Gases," July 1998. [Online]. Available: http://qginc.com/sites/default/files/Lawrence%20Berkeley%20National%20Laboratory%20%28LBNL%29%20Report%2040556.pdf. [Accessed 4 August 2015].

[27] M. Ahrens, "Smoke Alarms in U.S. Home Fires," March 2014. [Online]. Available: http://www.nfpa.org/research/reports-and-statistics/fire-safety-equipment/smoke-alarms-in-us-home-fires. [Accessed 4 August 2015].

[28] "Smoke Detectors - Ionization vs. Photoelectric," 26 February 2014. [Online]. Available: http://www.nfpa.org/safety-information/for-consumers/fire-and-safety-equipment/smoke-alarms/ionization-vs-photoelectric. [Accessed 4 August 2015].

[29] D. K. Roveti, "Choosing a Humidity Sensor: A Review of Three Technologies," Sensors Magazine, 1 July 2001. [Online]. Available: http://www.sensorsmag.com/sensors/humidity-moisture/choosing-a-humidity-sensor-a-review-three-technologies-840. [Accessed 2 July 2015].

[30] S. M. Sze and M. K. Lee, Semiconductor Devices, New York: John Wiley & Sons, 2012.

[31] G. Torres, "How Analog to Digital Converter Works," Hardware Secrets, 21 April 2006. [Online]. Available: http://www.hardwaresecrets.com/how-analog-to-digital-converter-adc-works/. [Accessed 14 July 2015].

[32] Coder-Tronics, "MSP 430 ACD Tutorial," 18 August 2013. [Online]. Available: http://coder-tronics.com/msp430-adc-tutorial/. [Accessed 14 July 2015].

[33] C. E. Willis, "User Interface Design for the Engineer," in *Electro/94 International*, 1994.

[34] J. Nielsen, "The difference between web design and GUI design," nngroup, 1 May 1997. [Online]. Available: http://www.nngroup.com/articles/the-difference-between-web-design-and-gui-design/. [Accessed 11 July 2015].

[35] IBM, "Internet of things," 23 June 2015. [Online]. Available: https://console.ng.bluemix.net/catalog/internet-of-things/. [Accessed 12 July 2015].

[36] Mozilla, "Web Developer Guide," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide. [Accessed 12 July 2015].

[37] PHP.net, "What is PHP?," [Online]. Available: http://php.net/manual/en/intro-whatis.php. [Accessed 12 July 2015].

[38] "How photoelectric smoke detectors work," Enggcyclopedia, [Online]. Available: http://www.enggcyclopedia.com/2011/11/photoelectric-smoke-detectors-work/. [Accessed 2 August 2015].

[39] A. V. Opeenheim and A. S. Willsky, Signals & Systems, New Delhi: Prentice-Hall of India, 2005.

[40] J. Salmela, "Raspberry Pi Webcam Over the Internet Using MJPG-Streamer," [Online]. Available: http://jacobsalmela.com/raspberry-pi-webcam-using-mjpg-streamer-over-internet/. [Accessed 2 August 2015].

[41] F. G. D. X. Y. T. Zhou Quan, "Trusted Architecture for Farmland Wireless Sensor Networks," 2012. [Online]. Available: http://ieeexplore.ieee.org.ezproxy.net.ucf.edu/stamp/stamp.jsp?tp=&arnumber=6427496. [Accessed 6 July 2015].