Wireless Home Control System

UCF

Spring - Summer 2015

Group #5

Grant Hernandez
Computer Engineer

Jimmy Campbell
Computer Engineer

Joseph Love
Electrical Engineer

Motivation

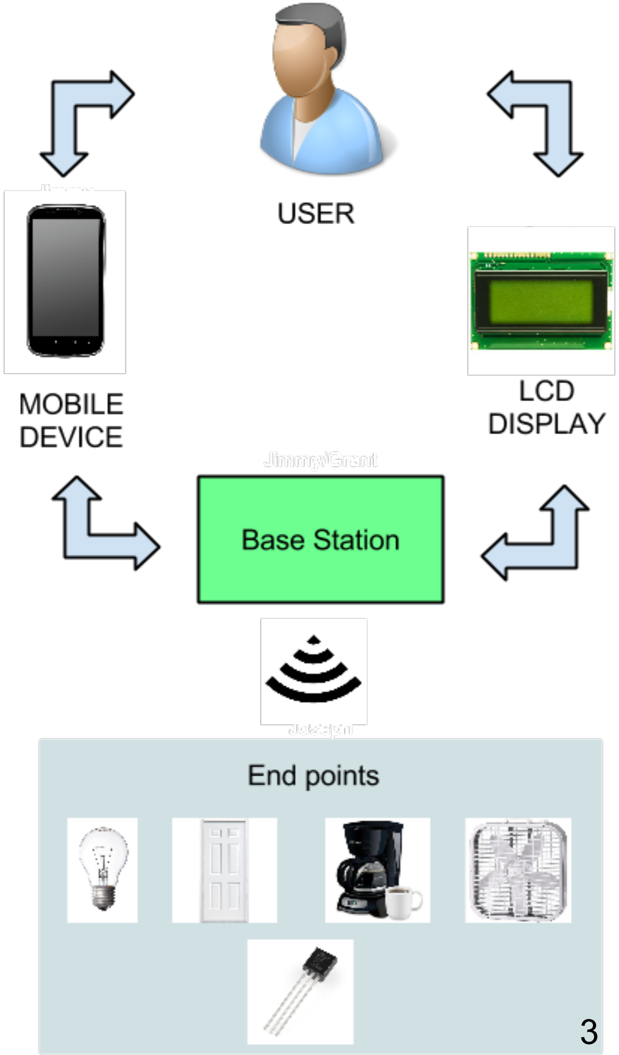
- Save time by automating mindless tasks
- Add to the comfort level of the home experience
- Create technology that can be useful to us personally in the future
- Enhance control and monitoring of the home system
- Compete with the growing home automation market

Goals and Objectives

- Complete control and monitoring of WHCS from an Android device
- Toggle and check status of lights and outlets
- Unlock and relock door and check door status
- Gather information from sensors placed in the home (motion, temperature, light, etc.)
- Check the power and connection status of each control module

System Overview

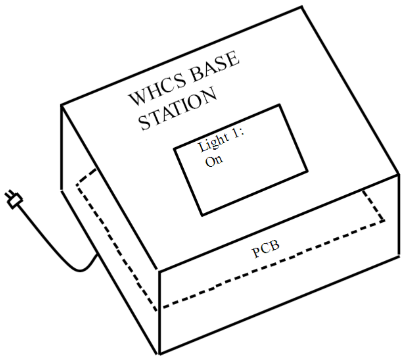
- User interacts with system within their home
- Begin interaction via LCD or Android phone
- Both of the available interfaces communicate with the Base Station
- Base Station maintains state of the system and delegates commands to control modules
- Control modules have the ability to interact with the home
- Control modules are specialized



Specifications and Requirements

- All in-house components suppliable through 120 VAC
- Radio communication must have at least a range of 50m
- Base station and control module interaction over 2.4 GHz radio
- BlueTooth communication must have at least a range of 15m
- Microcontroller must supply at least 30mA
- Microcontroller capable of achieving at least 8 MHz clock rate
- All logical components must be capable of operating in the 3.3V-5V range

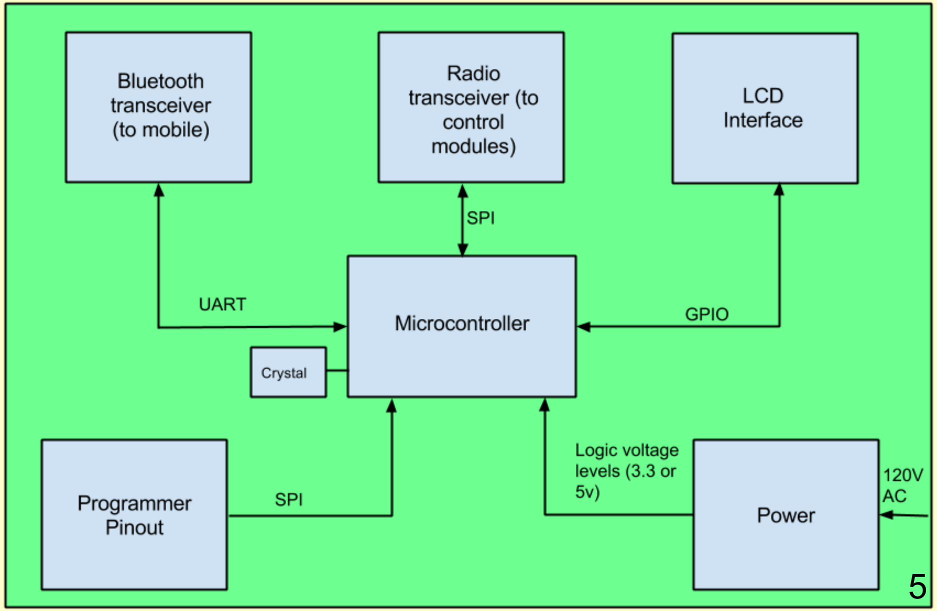
Base Station Block Diagram



- Central processing unit of WHCS
- Forwards commands from user to control modules, and updates from control module to user
- Uses an ATmega32A microcontroller due to high number of pins required and flash memory required
- Communicates with Android device through a BlueTooth module (HC-05) using the microcontroller's UART
- Communicates to control modules using a radio transceiver (NRF24L01+) connected through SPI

UART: Universal Asynchronous Receiver Transmitter
SPI: Serial Peripheral Interface
GPIO: General Purpose Input/Output

Base Station


Bluetooth transceiver (to mobile)
Radio transceiver (to control modules)
LCD Interface
SPI
UART
Microcontroller
GPIO
Crystal
Programmer Pinout
SPI
Logic voltage levels (3.3 or 5v)
Power
120V AC

Base Station Microcontroller

- The Base Station must have at least 27 free GPIO (includes SPI and UART)
- Based on these specifications, we chose the **ATmega32A**
  - Familiar ATmega328P (Arduino) doesn’t have the required pin count

Module	**Pins Required**
LCD	18
NRF24L01+	6
HC-05	3
TOTAL	**27**

ATmega32A Specifications	
Operating Voltage	2.7 - 5.5 V
Maximum Frequency	16 MHz
SRAM	2 KiB
Flash	32 KiB
Number of GPIO	**32**
Package	TQFP-44

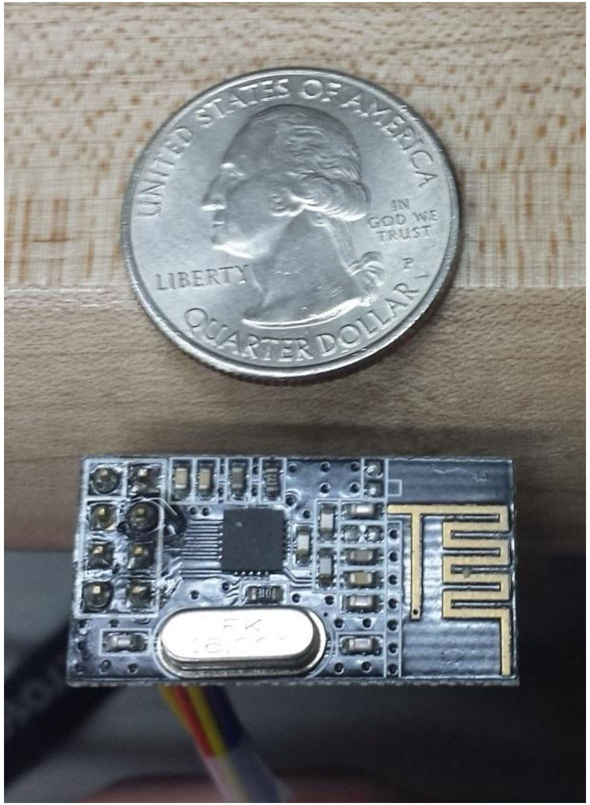
ATmega328P Specifications	
Operating Voltage	2.7 - 5.5 V
Maximum Frequency	20 MHz
SRAM	2 KiB
Flash	32 KiB
Number of GPIO	**23**
Package	TQFP-32

WHCS

NRF24L01+

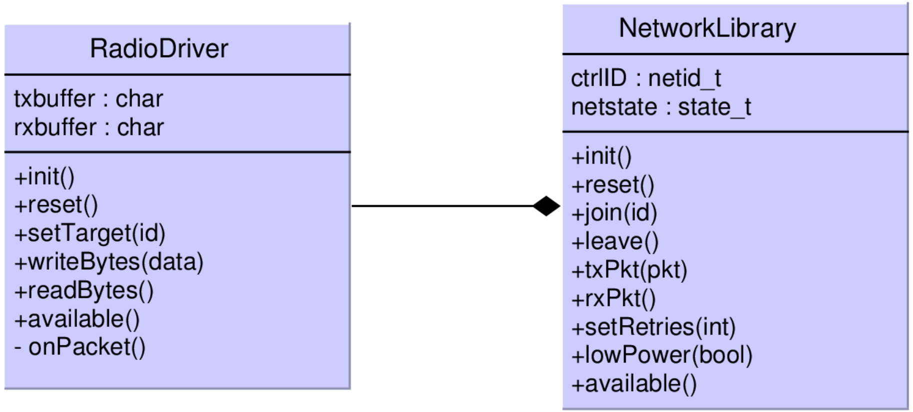
- The NRF is a radio transceiver that is used to provide the wireless communication across our boards
- Alternative to Wi-Fi, allows us to implement our own communication protocol and eliminate overhead of Wi-Fi
- Our data transfer is not enough to warrant the need of Wi-Fi, and the cost of Wi-Fi modules are more expensive
- Low power usage at 3.3V with 11.3 mA TX mode and 13.5 mA RX mode (from datasheet)
- Datasheet does not show max operating range but our tests show at least 50m, meeting specifications and more than enough for in house use.

Cost	Range (m)	Max Current Draw	Voltage	Size
$3.43	> 50	13.5mA	3.3V	15mm x 29mm



NRF24L01+ Driver

- Originally we were going to build our own NRF driver from the ground up and made good progress into this endeavour (We were able to transmit and perform basic operations with the NRF)
- We then realized that we could leverage a library made for Arduino (RF24) for the NRF24L01 by providing implementations for the Arduino specific function calls. The library is written in C++ so with Arduino code gone we got it to compile with `avr-gcc`
- Function calls like `digitalWrite()` which are Arduino specific had to be replaced, and we had to write the SPI library that plugs into the driver, but otherwise the C++ code was compatible.



RadioDriver

txbuffer : char
rxbuffer : char

+init()
+reset()
+setTarget(id)
+writeBytes(data)
+readBytes()
+available()
- onPacket()

NetworkLibrary

ctrlID : netid_t
netstate : state_t

+init()
+reset()
+join(id)
+leave()
+txPkt(pkt)
+rxPkt()
+setRetries(int)
+lowPower(bool)
+available()

BlueTooth Module

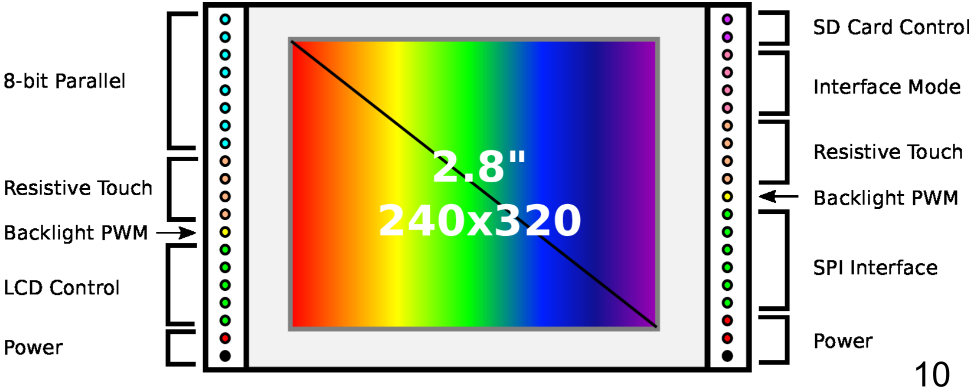
- Needed a method to communicate between base station and mobile, the solution was BlueTooth
- Research led to a choice between the RN-41 and HC-05 modules
- The RN-41 had better performance characteristics with an advertised range of 100m
- HC-05 is much cheaper and has a breakout option for easy prototyping
- Open field tests yielded a range of at least 50 meters for the HC-05 which met our specification guidelines
- Comparable size and performance led to choosing the cheaper part

	Cost	Range (m)	Break-out?	Configurable	Size (mm)
RN-41	$21.70	100	No	Yes	25.8 x 13.22
HC-05	$6.64	50+	Yes	Yes	27 x 13

LCD Controller

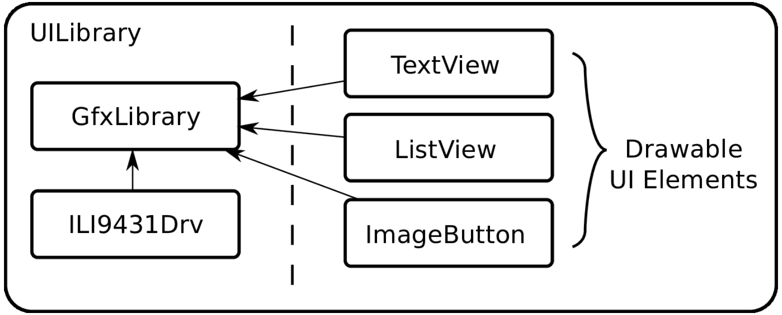
- Purchased from Adafruit
  - Added support for resistive touch screen
- ILI9341 TFT LCD Controller
  - Multiple MCU interfaces SPI or 8-bit parallel
  - We chose the 8-bit parallel interface for the speedup
  - Trade off: more microcontroller pins required
- Optional SD Card for images
  - Possible use for a high resolution logo

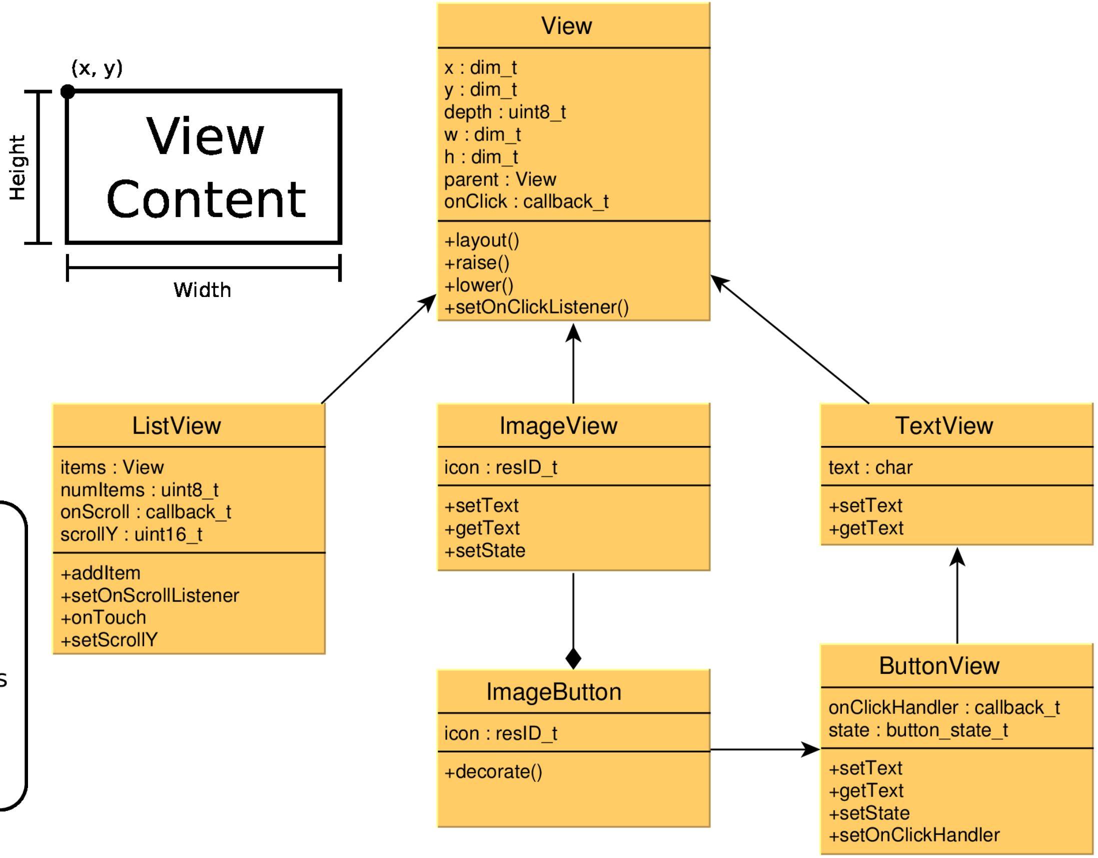
Specification	Description
Resolution	240x360
Colors	262K @ 18-bits, 65K @ 16bits
Voltage Input	3.3 - 5V
Weight	40 grams
Dimensions (just LCD)	2.8” diagonal
MCU Interface	Multiple. See Section 6.4.2
Touchscreen technology	Resistive (one finger)



UI Library

- View abstraction inspired by Android
  - Stores common attributes for all controls
- Derived controls specialize
- Perfect application of C++





View
x : dim_t
y : dim_t
depth : uint8_t
w : dim_t
h : dim_t
parent : View
onClick : callback_t
+layout()
+raise()
+lower()
+setOnClickListener()

ListView
items : View
numItems : uint8_t
onScroll : callback_t
scrollY : uint16_t
+addItem
+setOnScrollListener
+onTouch
+setScrollY

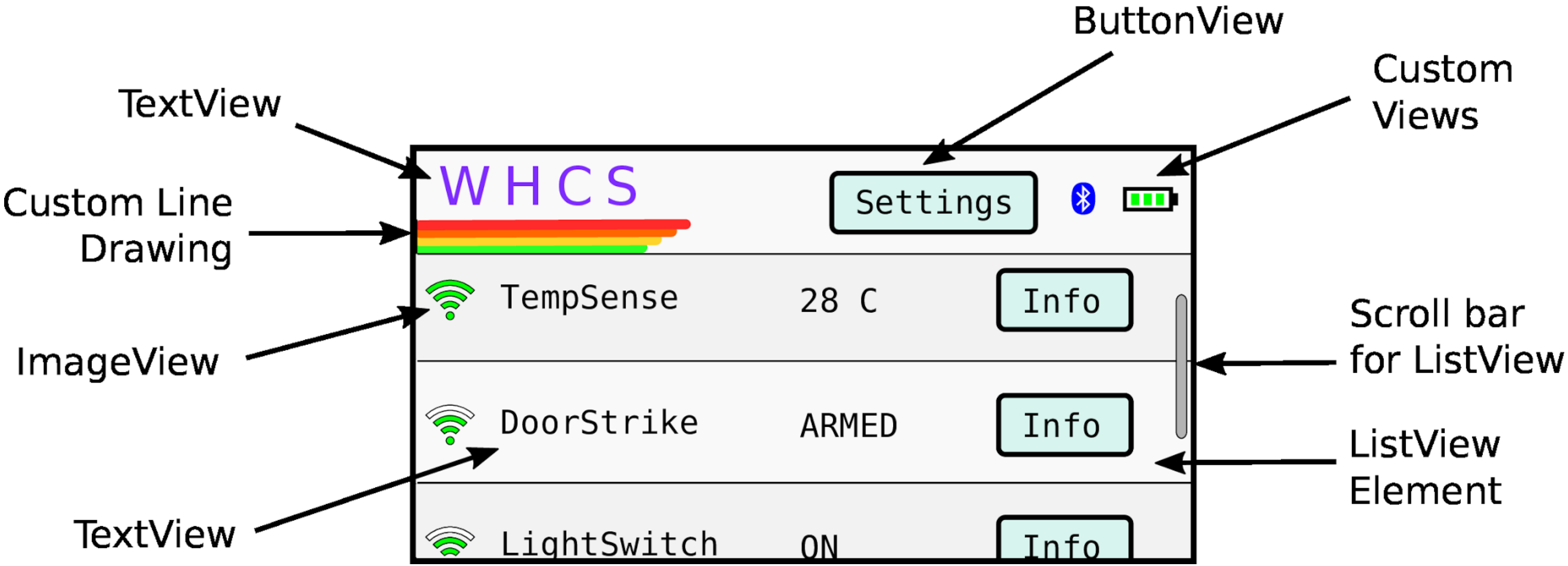
ImageView
icon : resID_t
+setText
+getText
+setState

TextView
text : char
+setText
+getText

ImageButton
icon : resID_t
+decorate()

ButtonView
onClickHandler : callback_t
state : button_state_t
+setText
+getText
+setState
+setOnClickHandler

Base Station UI Mockup



TextView

Custom Line Drawing

ImageView

TextView

ButtonView

Custom Views

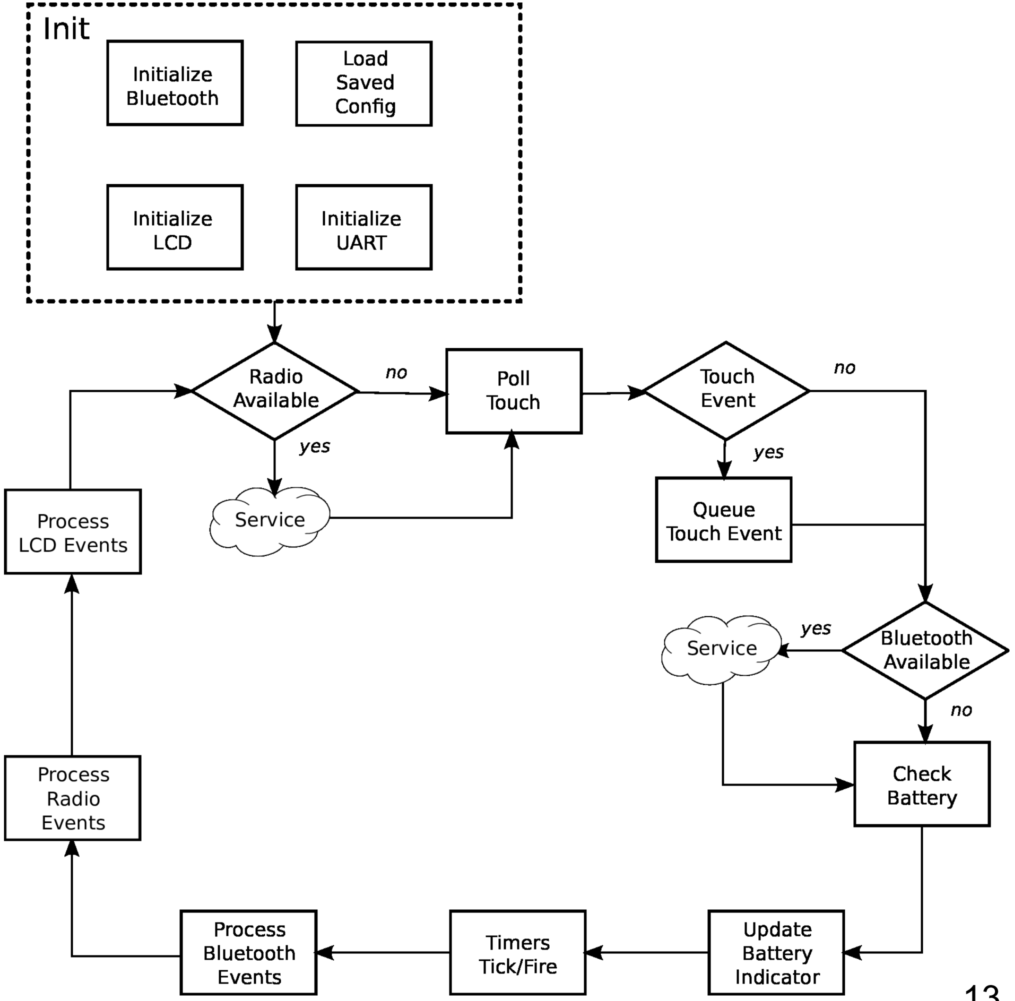
Scroll bar for ListView

ListView Element

A potential touch interface for the Base Station LCD

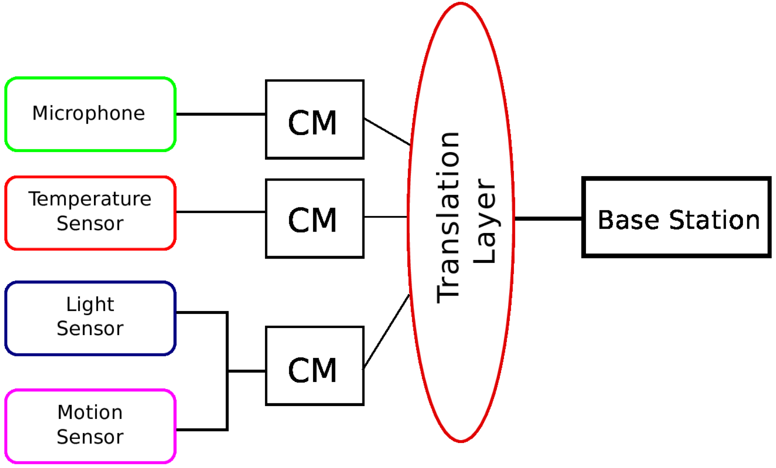
Base Station Software Loop

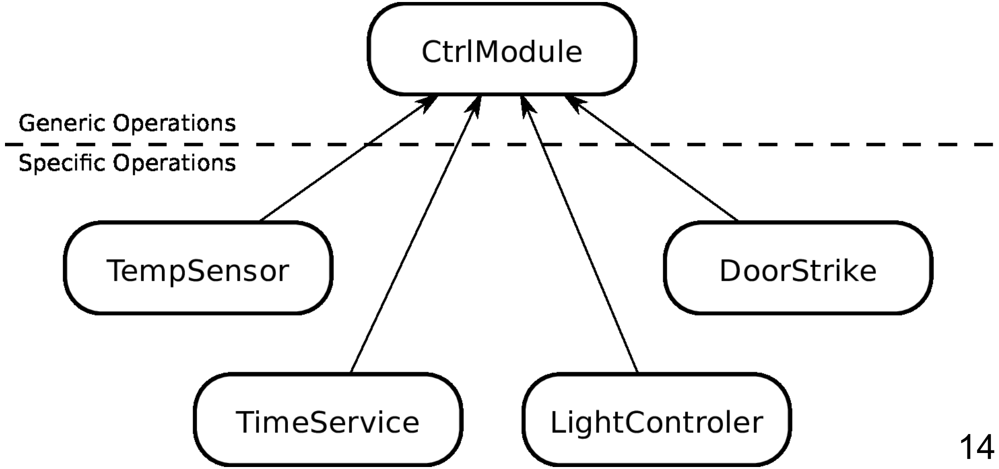
- Initialize the major components
- Service radios
- Poll for touch events
- Redraw dirty LCD regions
- Generate radio responses and actions from received commands/data
- Tick global timers
- Fire events on expired timers

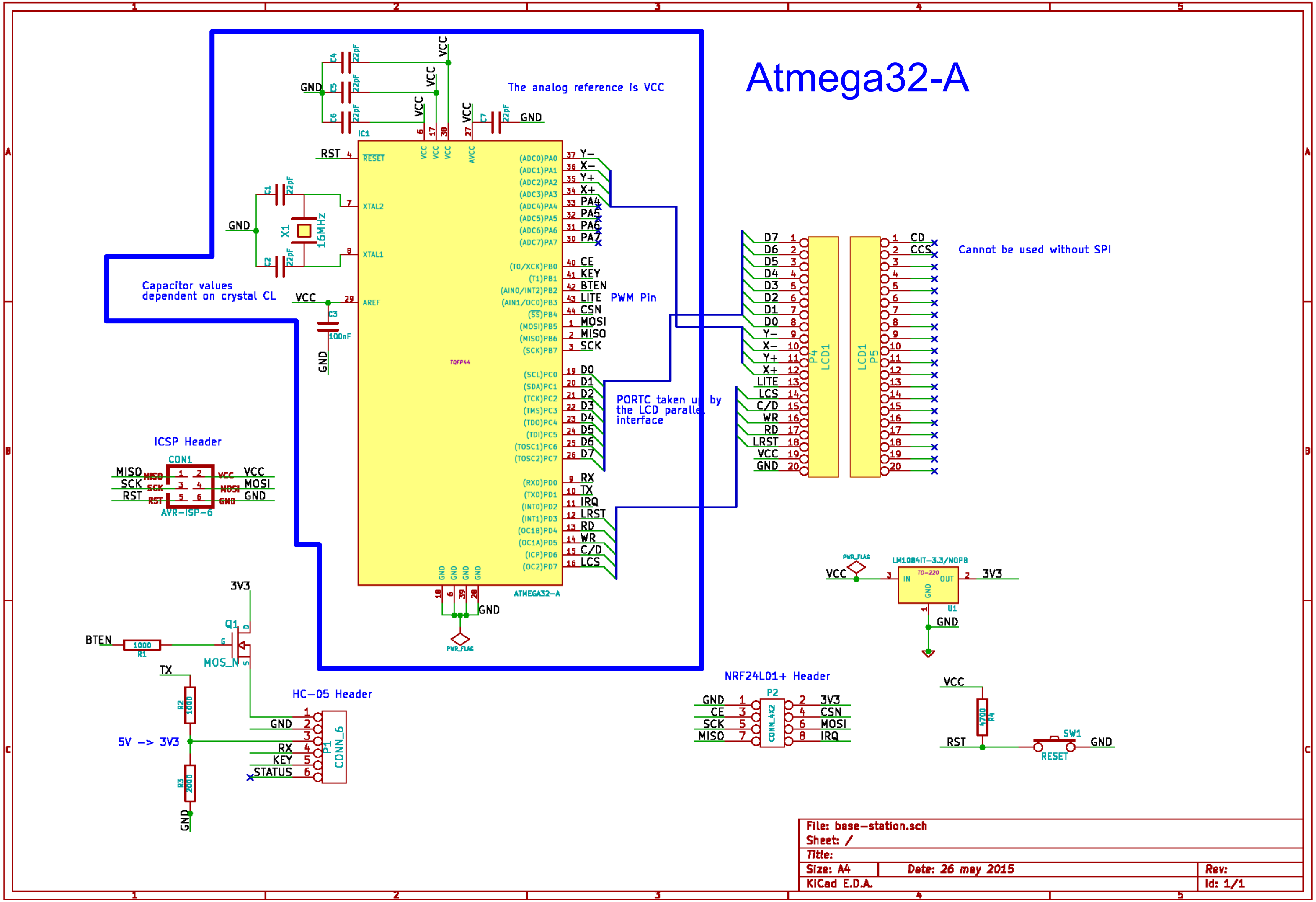


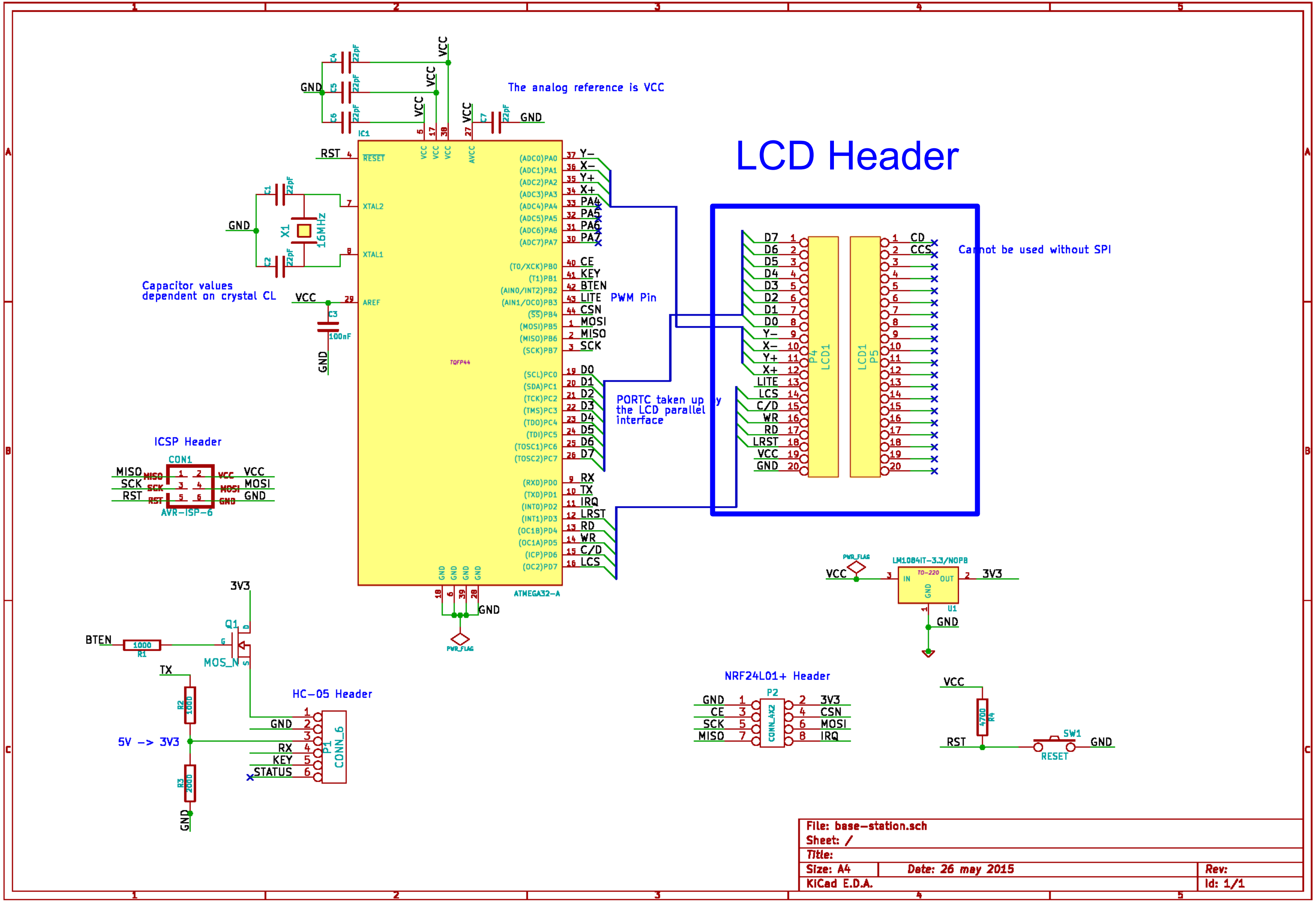
Control Module Abstraction

- In order for WHCS to scale with many control modules, we have designed the network around a generic control module
- This module will have specific role a specific functions for sending and receiving data packets



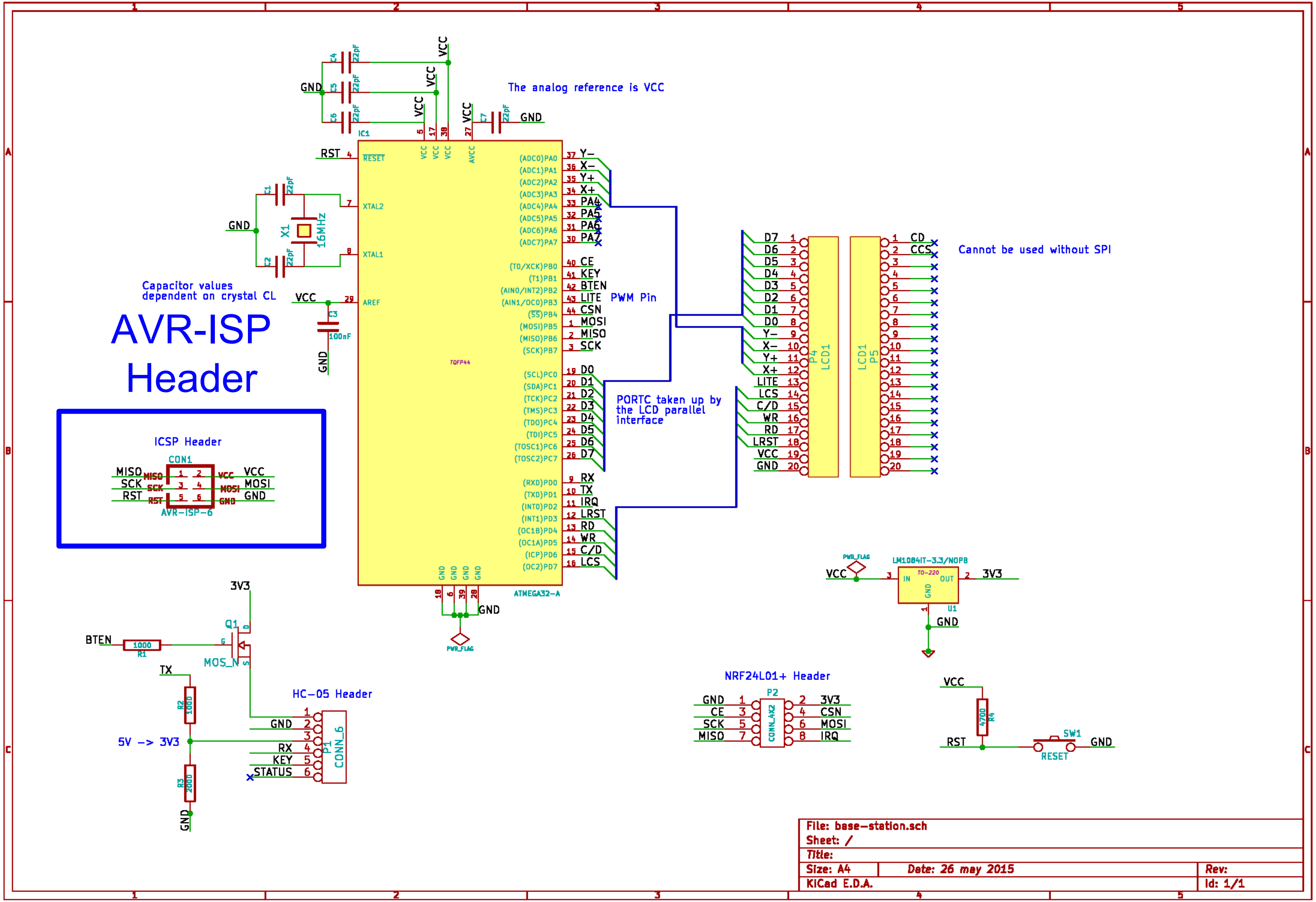



Atmega32-A
The analog reference is VCC
Capacitor values dependent on crystal CL
PWM Pin
PORTC taken up by the LCD parallel interface
Cannot be used without SPI
ICSP Header
CON1
AVR-ISP-6
MISO
SCK
RST
VCC
MOSI
GND
HC-05 Header
P1
CONN_6
GND
RX
KEY
STATUS
5V -> 3V3
R1 1000
R2 1000
R3 2000
Q1
MOS_N
BTEN
TX
3V3
NRF24L01+ Header
P2
CONN_4X2
CE
CSN
IRQ
LM1084IT-3.3/NOPB
TO-220
U1
PWR_FLAG
IC1
ATMEGA32-A
TQFP44
X1 16MHz
C1 22pF
C2 22pF
C3 100nF
C4 22pF
C5 22pF
C6 22pF
C7 22pF
RESET
XTAL2
XTAL1
AREF
AVCC
P4 LCD1
LCD1 P5
D7
D6
D5
D4
D3
D2
D1
D0
Y-
X-
Y+
X+
LITE
LCS
C/D
WR
RD
LRST
CD
CCS
R4 4700
SW1
RESET
File: base-station.sch
Sheet: /
Title:
Size: A4
Date: 26 may 2015
Rev:
KiCad E.D.A.
Id: 1/1

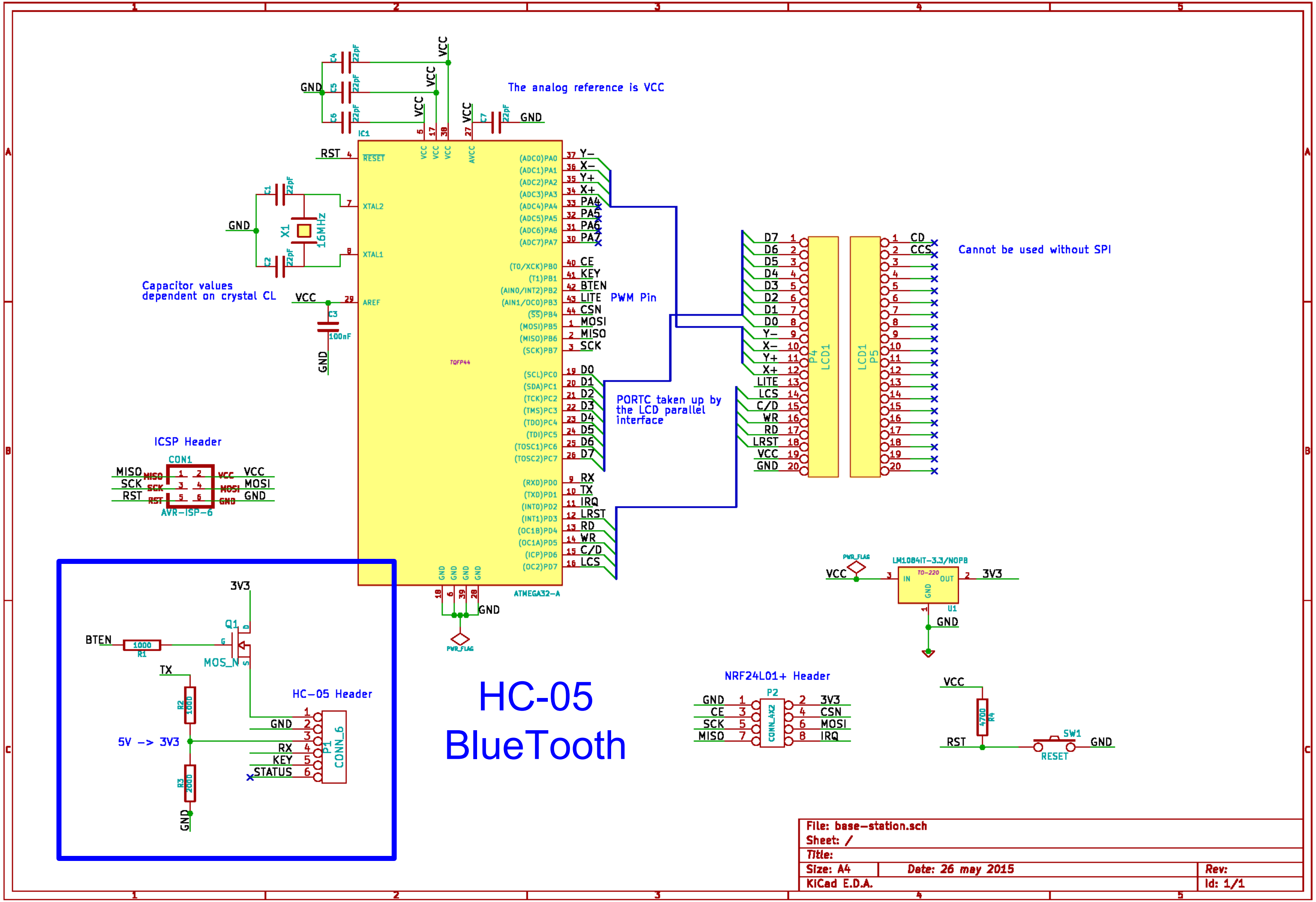


LCD Header

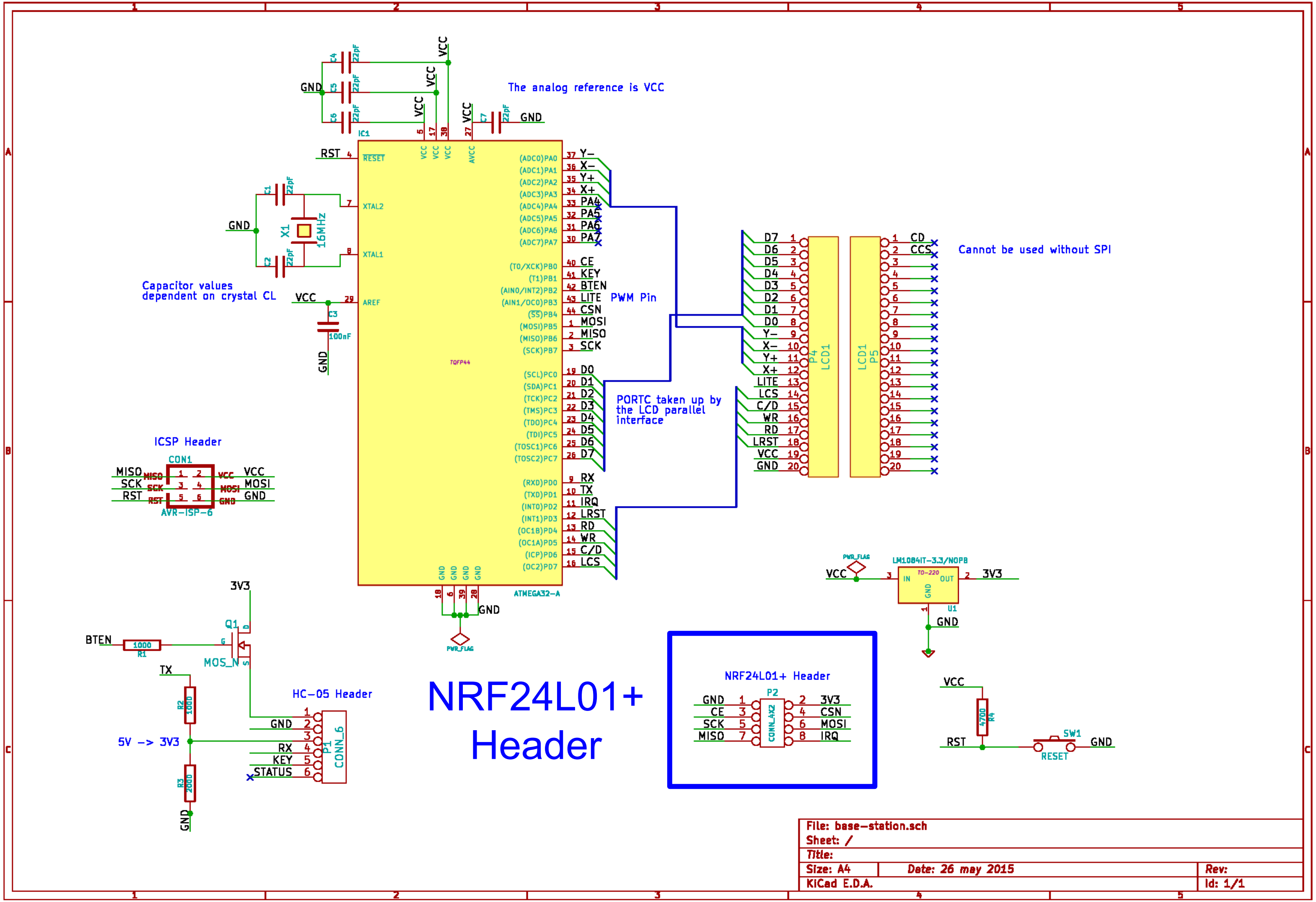
The analog reference is VCC
Capacitor values dependent on crystal CL
PWM Pin
PORTC taken up by the LCD parallel interface
Cannot be used without SPI
ICSP Header
CON1
AVR-ISP-6
MISO
SCK
RST
VCC
MOSI
GND
HC-05 Header
P1
CONN_6
GND
RX
KEY
STATUS
5V -> 3V3
BTEN
TX
3V3
Q1
MOS_N
R1 1000
R2 1000
R3 2000
NRF24L01+ Header
P2
CONN_4X2
GND
CE
SCK
MISO
3V3
CSN
MOSI
IRQ
LM1084IT-3.3/NOPB
TO-220
U1
IN
OUT
PWR_FLAG
R4 4700
SW1
RESET
IC1
ATMEGA32-A
TQFP44
X1 16MHz
C1 22pF
C2 22pF
C3 100nF
C4 22pF
C5 22pF
C6 22pF
C7 22pF
XTAL2
XTAL1
AREF
AVCC
Y-
X-
Y+
X+
PA4
PA5
PA6
PA7
CE
KEY
BTEN
LITE
CSN
MOSI
MISO
SCK
D0
D1
D2
D3
D4
D5
D6
D7
RX
TX
IRQ
LRST
RD
WR
C/D
LCS
P4
LCD1
P5
CD
CCS
LITE
LCS
C/D
WR
RD
LRST
File: base-station.sch
Sheet: /
Title:
Size: A4
Date: 26 may 2015
Rev:
KiCad E.D.A.
Id: 1/1



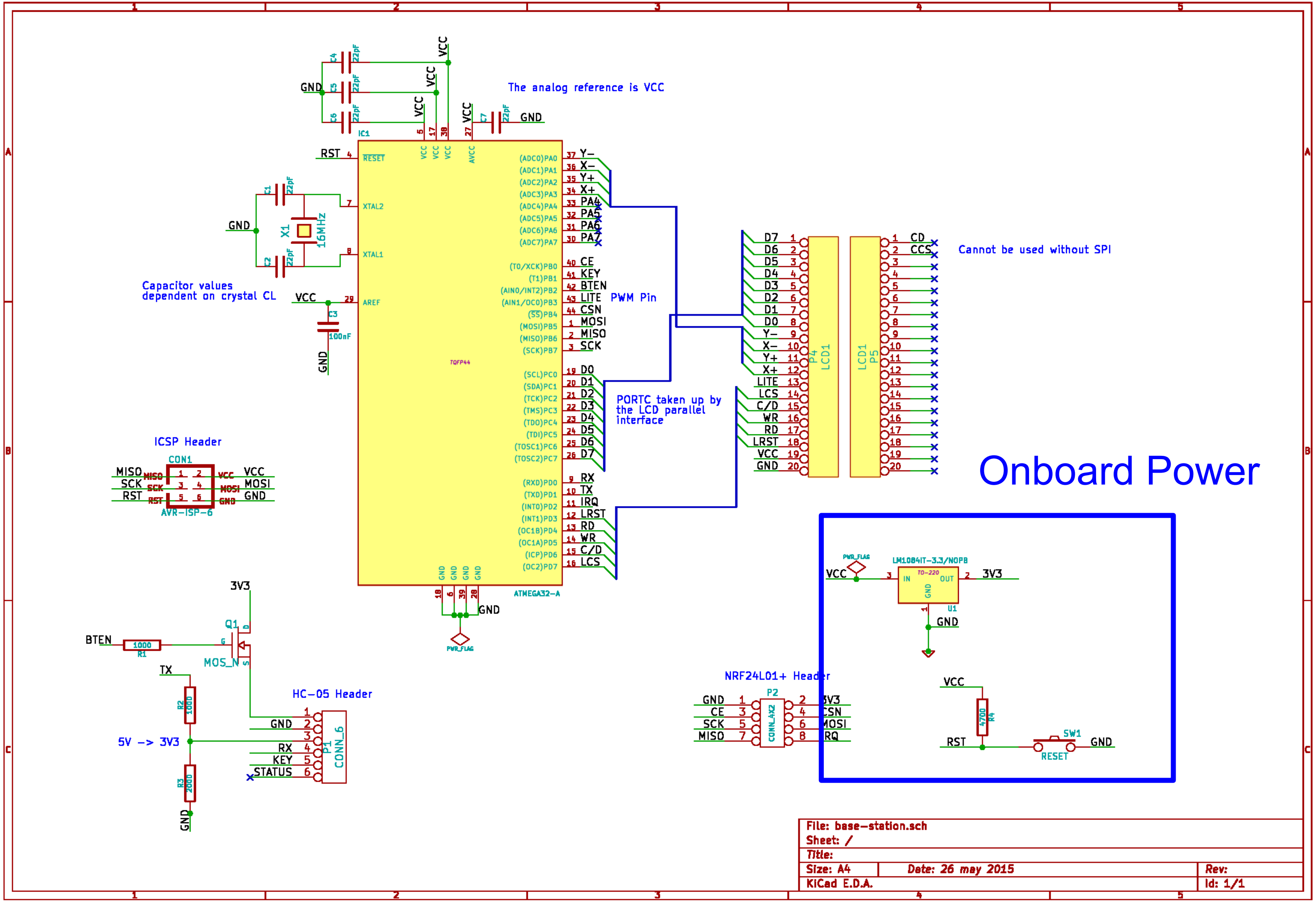
File: base-station.sch
Sheet: /
Title:
Size: A4 Date: 26 may 2015 Rev:
KiCad E.D.A. Id: 1/1



HC-05 BlueTooth



NRF24L01+ Header



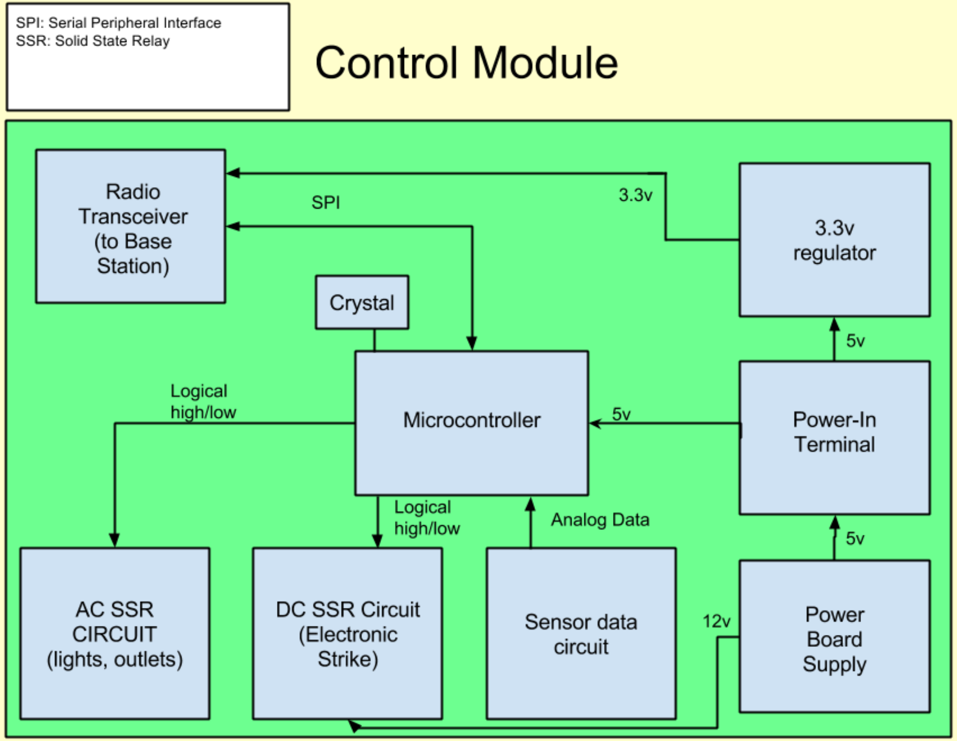
Onboard Power

File: base-station.sch
Sheet: /
Title:
Size: A4 Date: 26 may 2015 Rev:
KiCad E.D.A. Id: 1/1

WHCS

Control Module Block Diagram

- Features a modular design where any control module can serve any purpose
- Only the parts needed for the control module’s role in WHCS need be populated
- Receives power from the WHCS power board and downsteps the 5v to 3.3v for radio transceiver
- Features three main logic circuits for interacting with WHCS targets
  - AC switching circuit
  - DC switching circuit
  - Analog sensor circuit



SPI: Serial Peripheral Interface
SSR: Solid State Relay

Control Module

Control Module Microcontroller

- Unlike the base station, the Control Module only manages an NRF radio and some assorted peripherals
- Therefore, for design simplicity and to stay with the same family of MCUs, we chose the **ATmega328P** for the Control Module

Module	**Pins Required**
Assorted Ctrl.	3
NRF24L01+	6
TOTAL	**9**

ATmega328P Specifications	
Number of GPIO	**23**

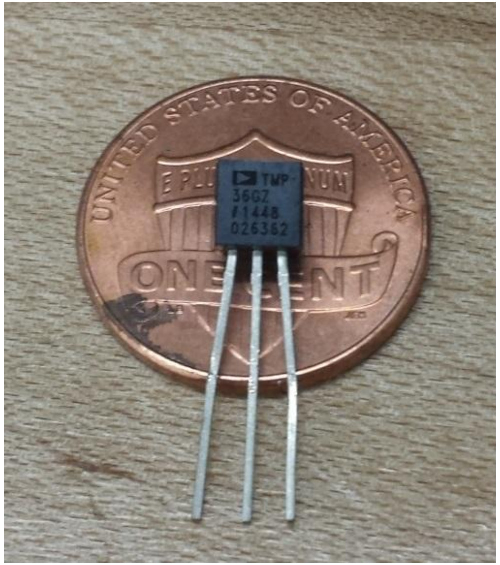
Control Module Relays

- Relays needed to switch AC outlet, lights, and the electronic strike
- Wanted relays that could be activated directly from the mcu
- Microcontroller can supply at most 50 mA
- Typical mechanical relays draw too much current & voltage requirements are too high
- Solid state relays allow us to control our high power circuits directly from the microcontroller
- We picked the cheapest solid state relays that met our power needs

Solid State Relay	**Forward Voltage**	**Load Voltage**	**Load Current**
S116S02F	1.2V DC	120V AC	16A
CPC1002NTR	1.2V DC	12V DC	700mA

Relay Type	**Voltage Rating**	**Current Rating**	**Activation Voltage**	**Activation Current**
Solid State	> 120 VAC	15A	1.2V	20 mA
Mechanical	> 120 VAC	15A	3.5V	84 mA (too high)

Temperature Sensor



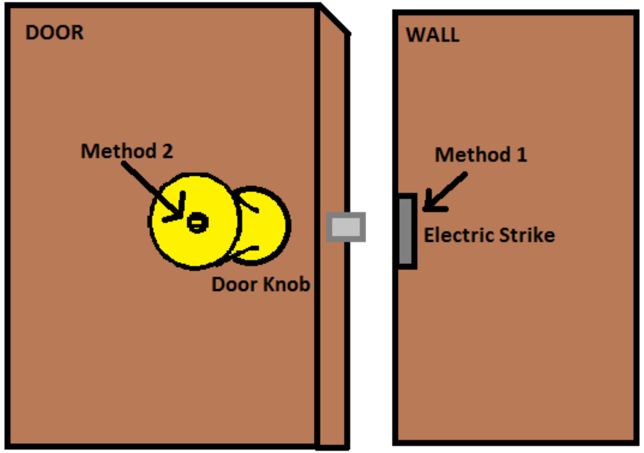
- The WHCS control module design provides support for analog sensors
- For our prototype we will be using a temperature sensor
- Fine accuracy of temperature in home is not critical
- Focus on temperature sensor was low cost, ease of use, and analog capability
- The **TMP36** is a popular analog temperature sensor with accuracies of +-2°C from -40°C to 125°C
- This part satisfies our requirements

Operating Voltage	**Supply Current**	**Temperature Range**
2.7V to 5.5V	< 50µA	-40°C to 125°C

WHCS

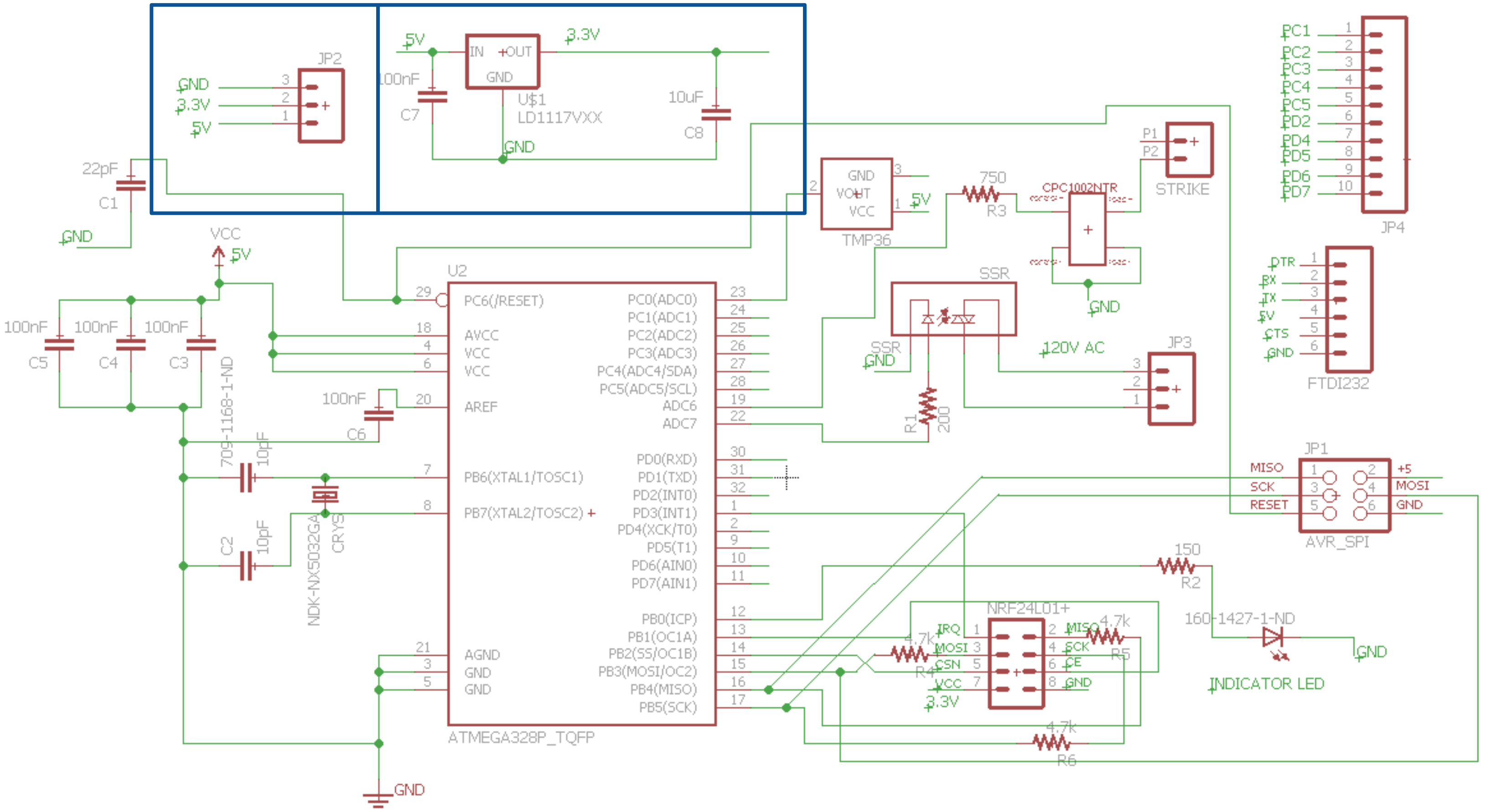
Access control

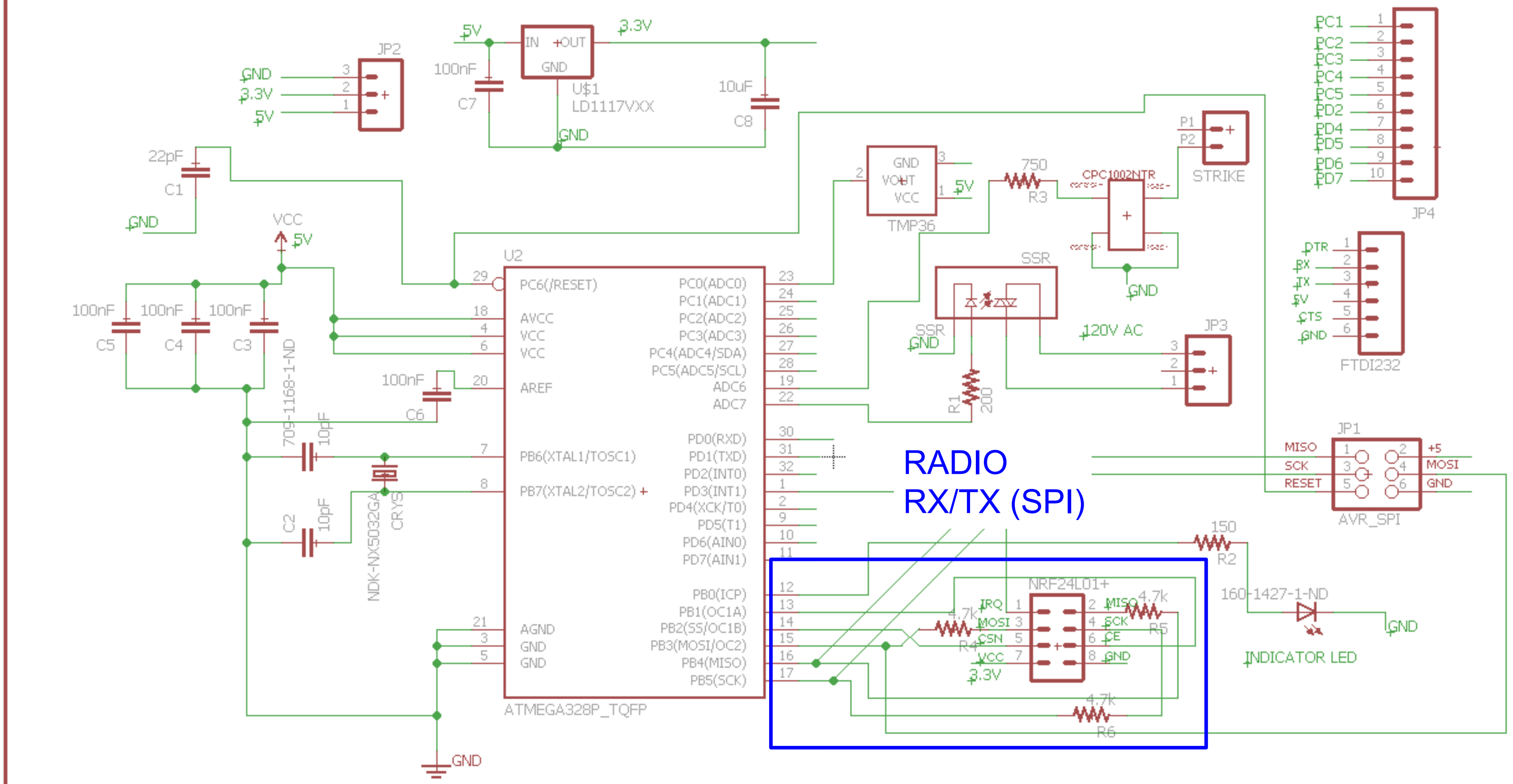
- Normally open electronic strike used for access control so that door is normally locked until supplied power
- Manual lock still used as a secondary method of entry in case of power failure
- 1X ELSTRAB5NO strike operates at 12V and draws 450mA



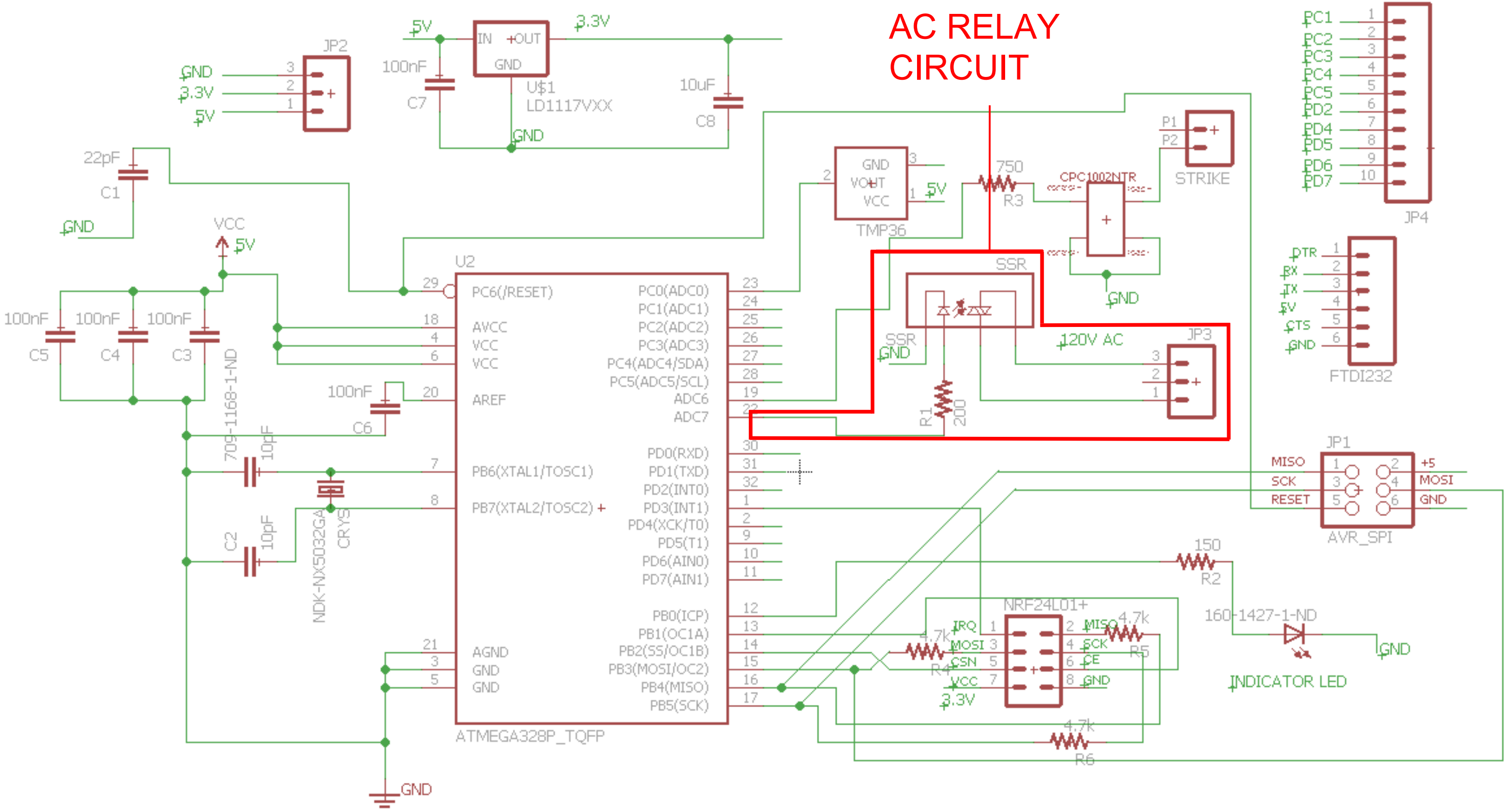
Power In

3.3V Regulator

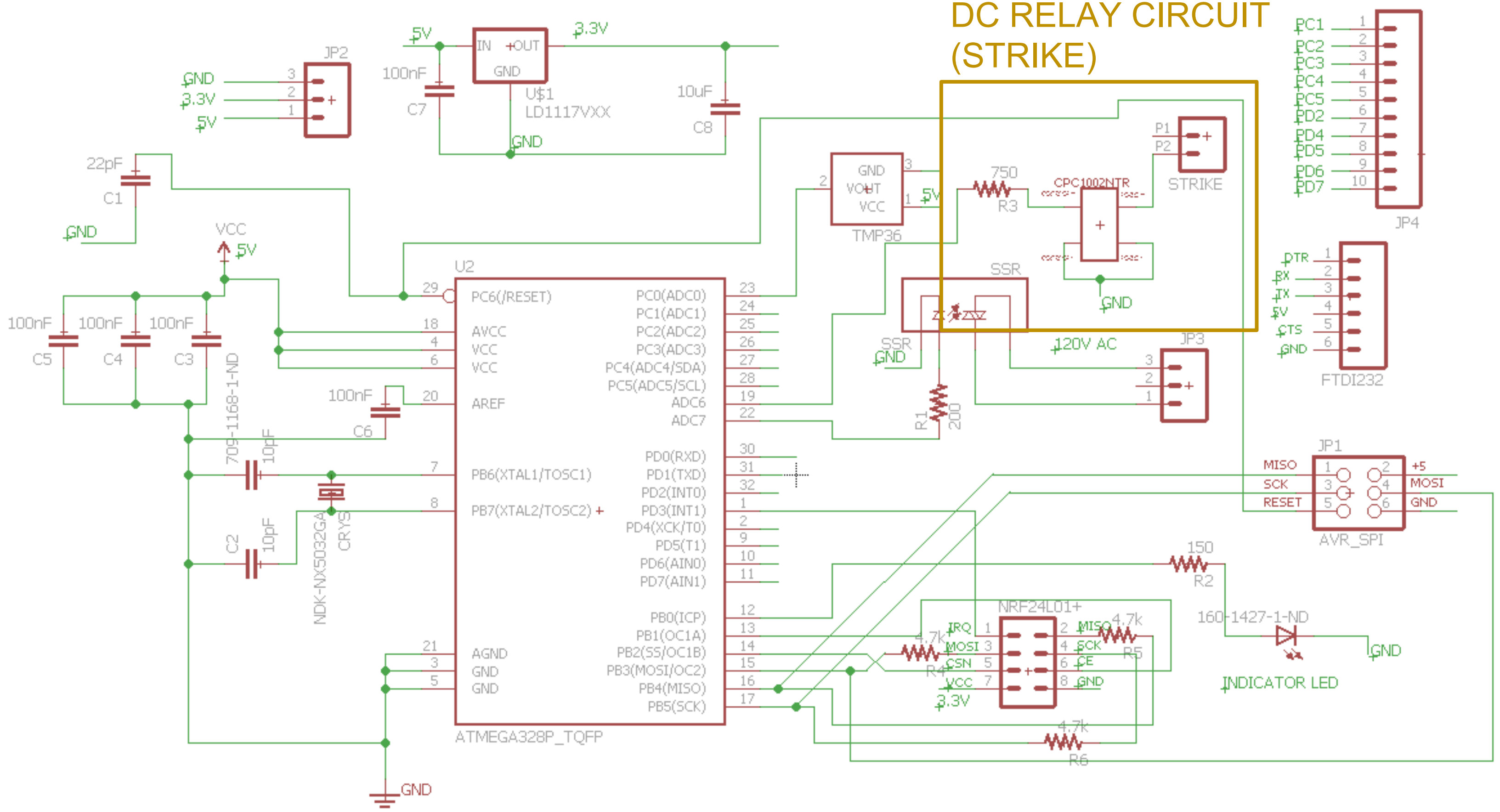




RADIO
RX/TX (SPI)

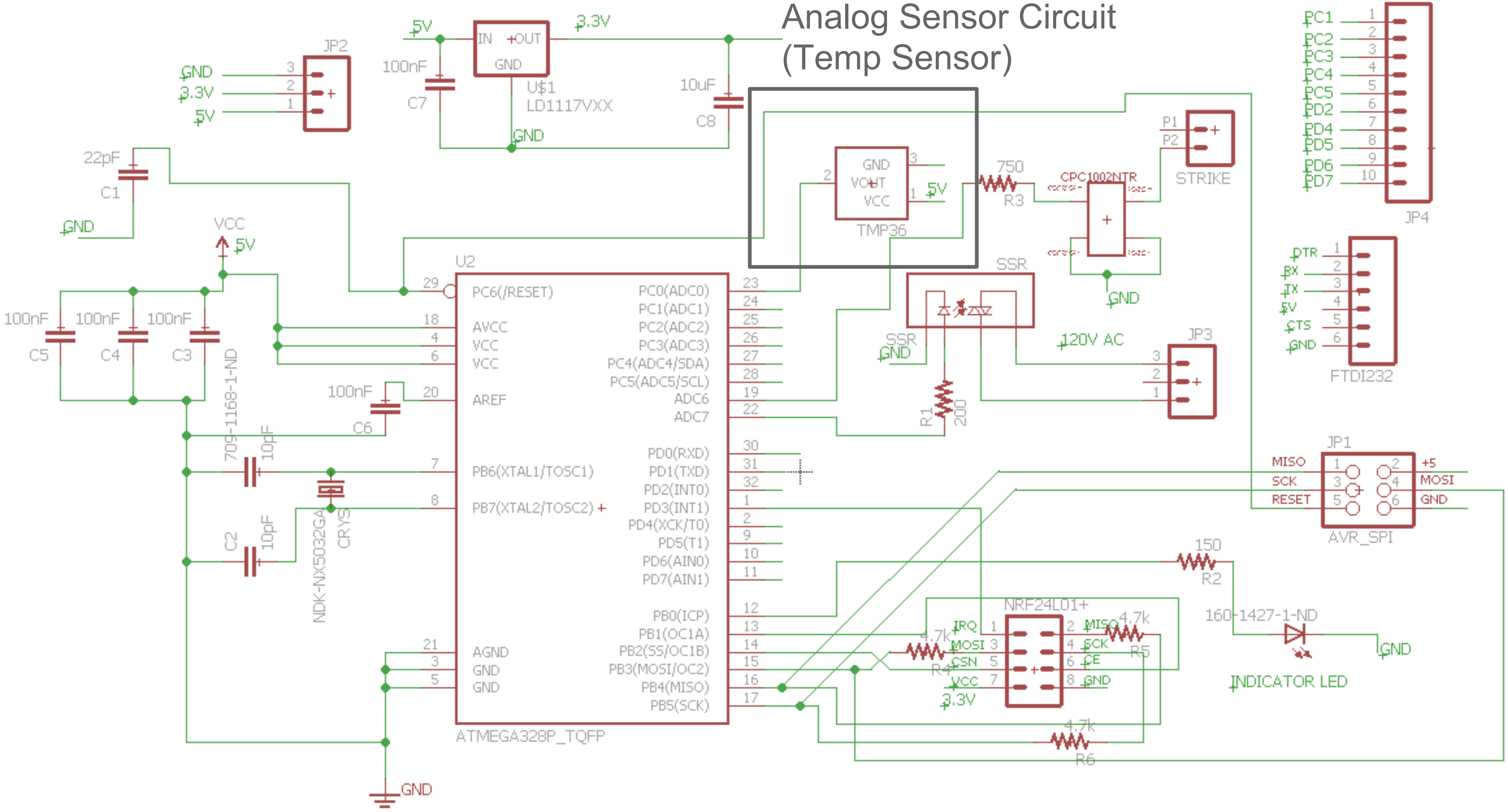


AC RELAY CIRCUIT



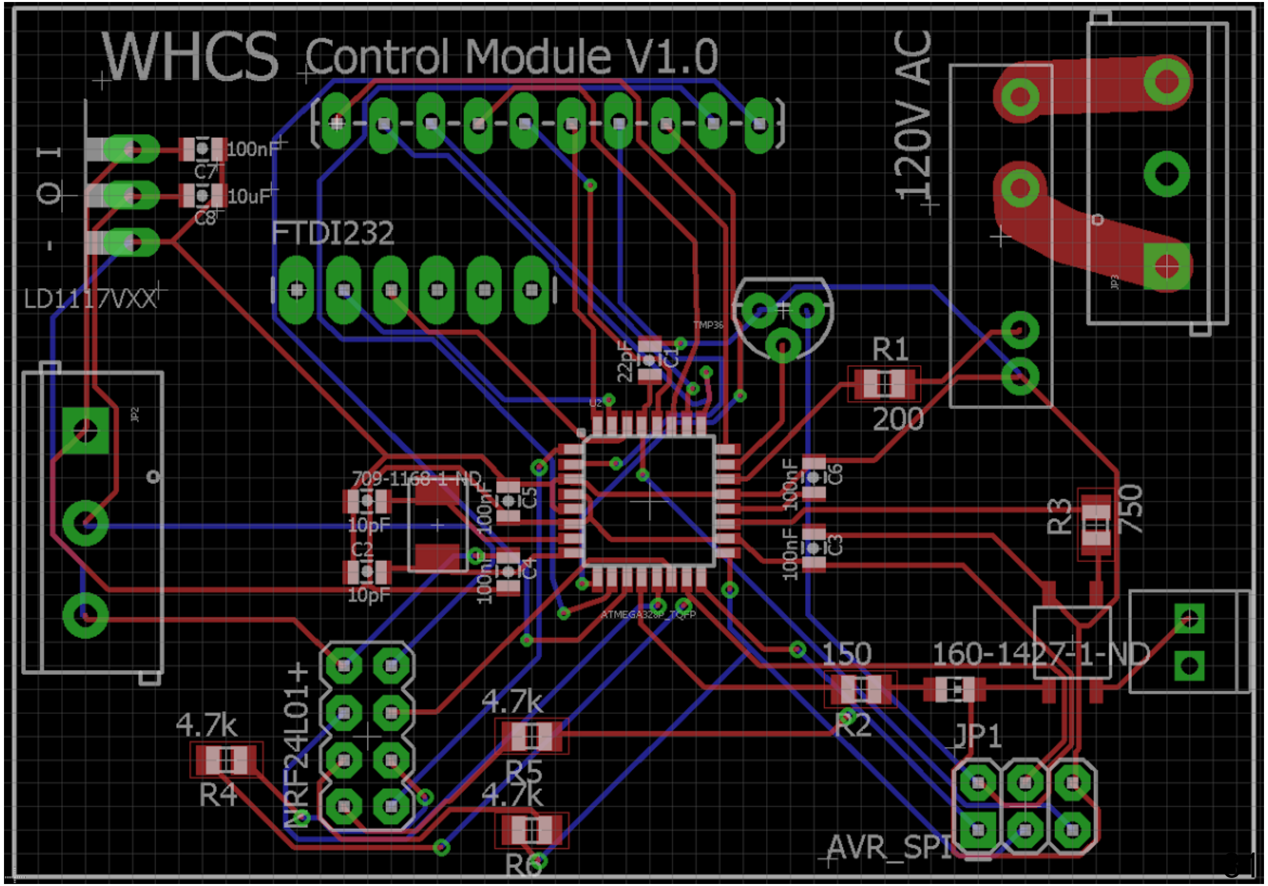
DC RELAY CIRCUIT (STRIKE)

Analog Sensor Circuit (Temp Sensor)



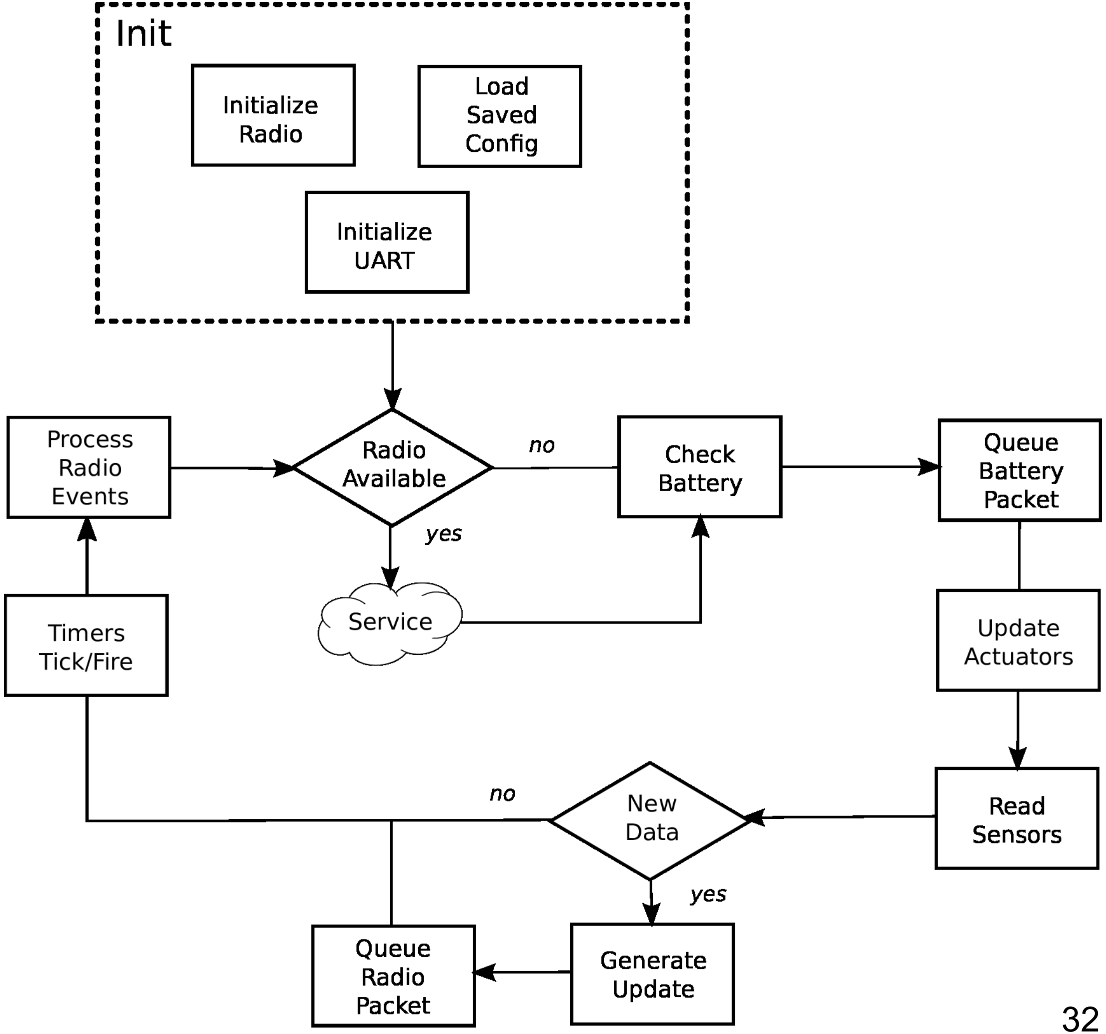
Control Module Board Layout

- Designed in Eagle
- Used existing footprints when available (SparkFun library)
- Used reference schematics when available (Arduino reference design for external crystal)
- Used trace width calculator to ensure that trace widths were safe for the amperage allowable in the AC relay
- Chose 0805 resistor and capacitor size for ease of soldering in prototype phase
- Order placed at OSH park for fabrication



Control Module Software Loop

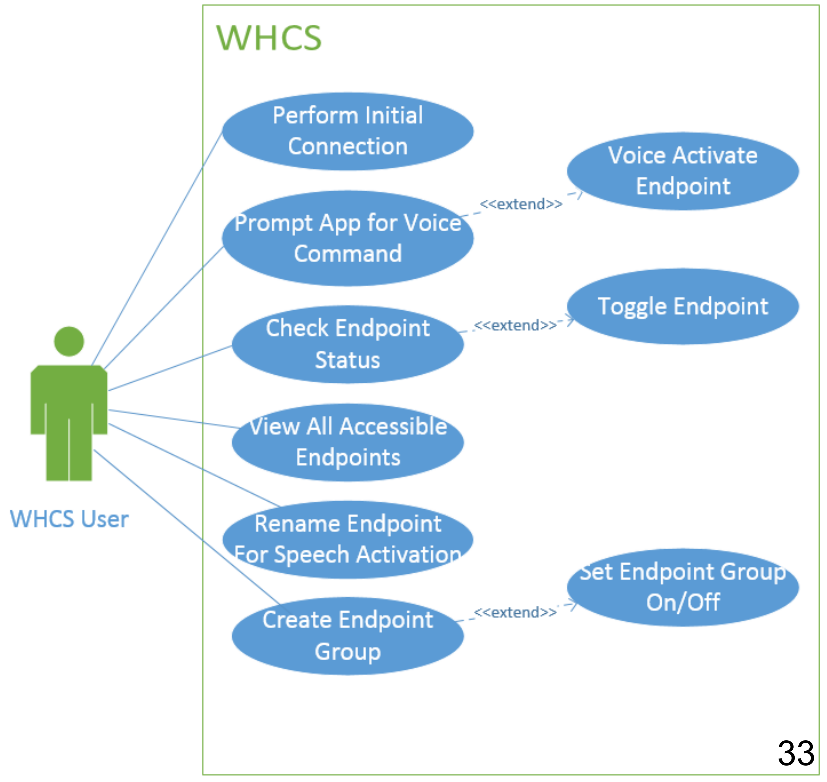
- Initialize major components
- Check for radio events
- Generate actions from received radio events
- Tick timers and run timer callbacks
- Process events and update the state of any switches



WHCS

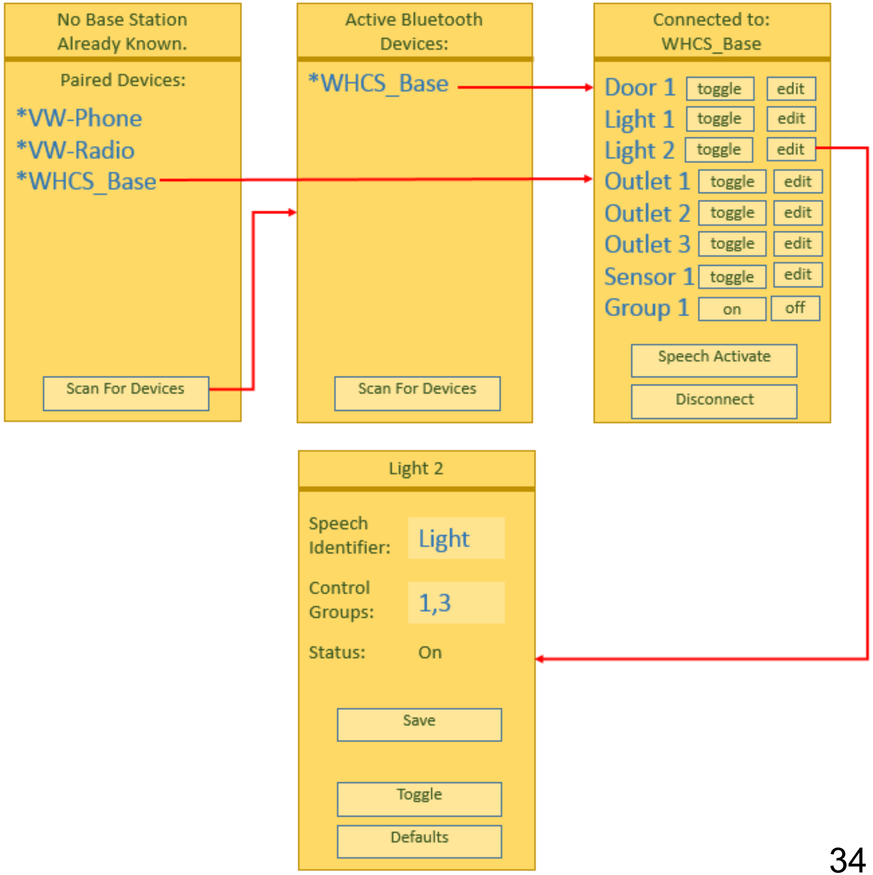
Android Use Case Diagram

- WHCS provides intuitive and fluid use cases
- High priority cases
  - Connect to WHCS
  - Change system state
  - Query system state
  - Perform voice command from Android
- Extended feature cases
  - Create endpoint groups
  - Change speech identifier of endpoint


WHCS
Perform Initial Connection
Voice Activate Endpoint
<<extend>>
Prompt App for Voice Command
Toggle Endpoint
<<extend>>
Check Endpoint Status
View All Accessible Endpoints
WHCS User
Rename Endpoint For Speech Activation
Set Endpoint Group On/Off
<<extend>>
Create Endpoint Group

GUI Mock Flow

- WHCS Android application will follow Android Material design guidelines
- The first piece of the application users will encounter is connecting to the base station
- Need BlueTooth enabled, need to select Base Station
- Once connected to Base Station, displays all control modules present in system
- From Control Module list can toggle state in system or select control module to observe state
- In Control Module detail view can change name and speech identifier for control module



Speech Identifier: Light

Control Groups: 1,3

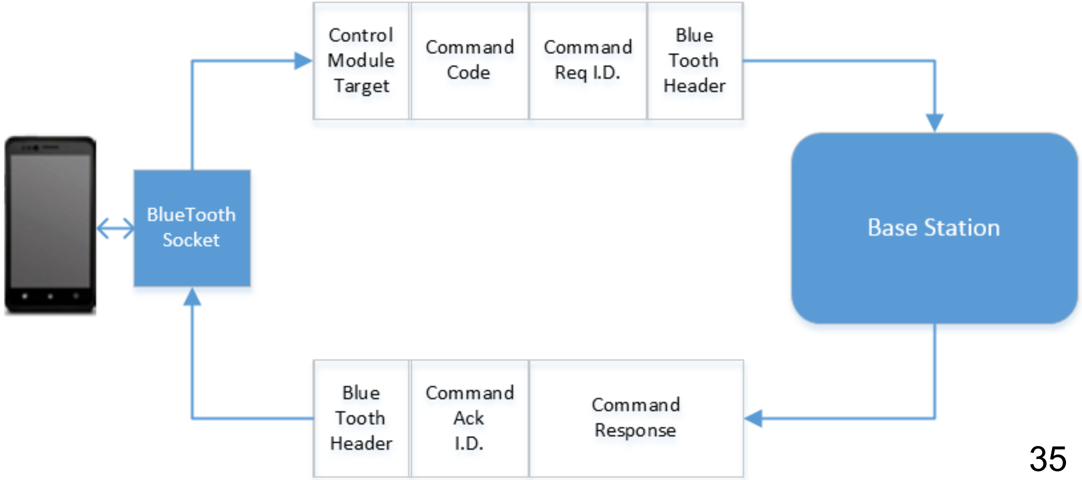
Status: On

WHCS

Android to Base Station Protocol

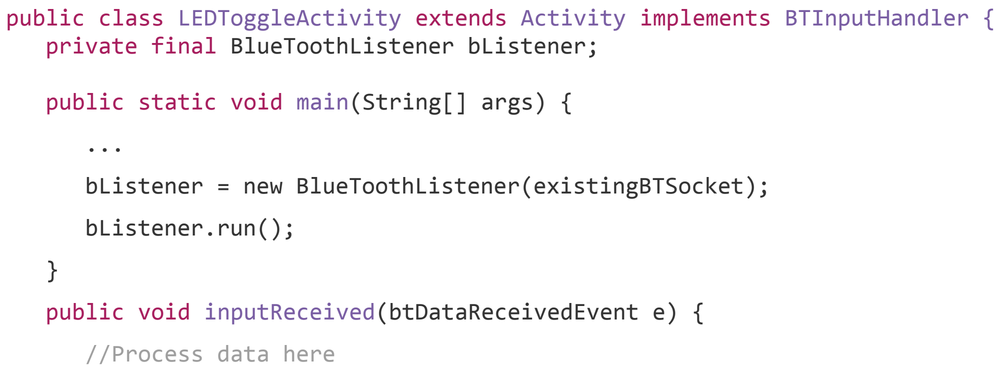
Command Req. ID	Command Code	Control Mod. Tgt
00000000	00000001	00000100
0th command	Toggle	4th Module

- The Base Station always acts as a slave device
- Android issues commands over the BlueTooth link
- Each command has a command identifier that way the status of the command can be tracked through the execution process
- The Command Code field acts as an Op-Code in a microcontroller
- Most commands will be delegated to a specific control module specified in the control module target field



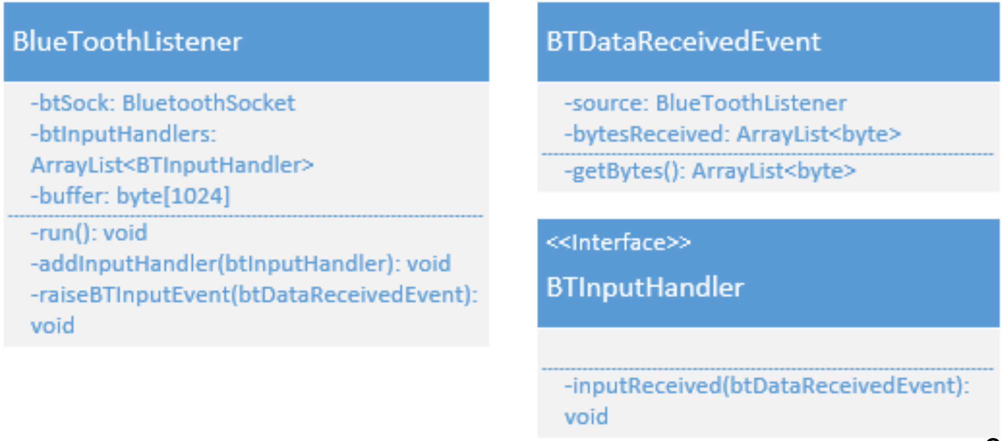
WHCS BlueToothListener

- After issuing a command, the app will have to await response
- Waiting cannot be done on main thread which handles UI
- Solution is custom event based receiving class “BlueToothListener”
- The application creates the BlueToothListener on startup and it raises an event whenever data is received over the BlueTooth socket
- The application can respond to the data received in whatever way necessary and does not have to block



```
public class LEDToggleActivity extends Activity implements BTInputHandler {
    private final BlueToothListener bListener;

    public static void main(String[] args) {
        ...
        bListener = new BlueToothListener(existingBTSocket);
        bListener.run();
    }
    public void inputReceived(btDataReceivedEvent e) {
        //Process data here
```



BlueToothListener

-btSock: BluetoothSocket
-btInputHandlers: ArrayList<BTInputHandler>
-buffer: byte[1024]

-run(): void
-addInputHandler(btInputHandler): void
-raiseBTInputEvent(btDataReceivedEvent): void

BTDataReceivedEvent

-source: BlueToothListener
-bytesReceived: ArrayList<byte>

-getBytes(): ArrayList<byte>

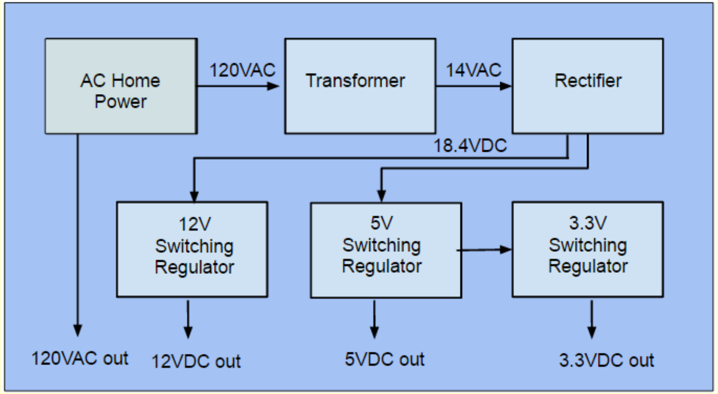
<<Interface>>
BTInputHandler

-inputReceived(btDataReceivedEvent): void

WHCS

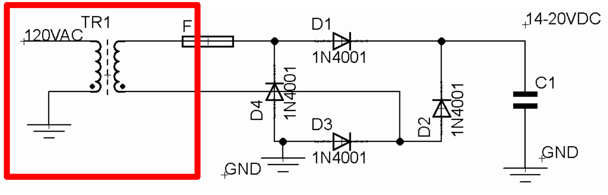
Power Board Block Diagram

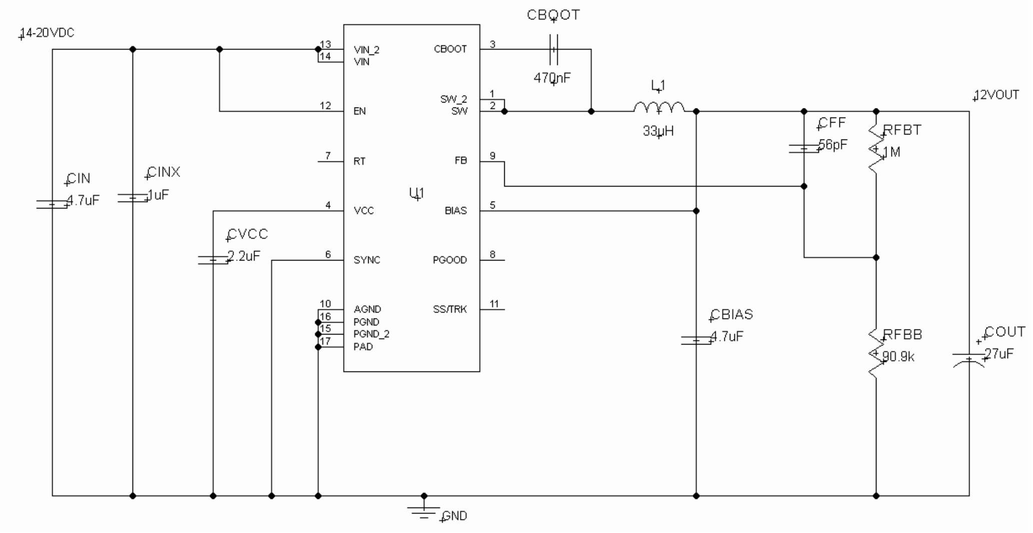
- WHCS will be completely powered by the home
- Board design will provide lines of 120VAC, 12VDC, 5VDC, 3.3VDC
- Populate differently depending on the board it supplies power to.
- 5V and 3.3V line designed separately from the 12V line
  - 12V only needed for one control module while 3.3V and 5V is used in each board

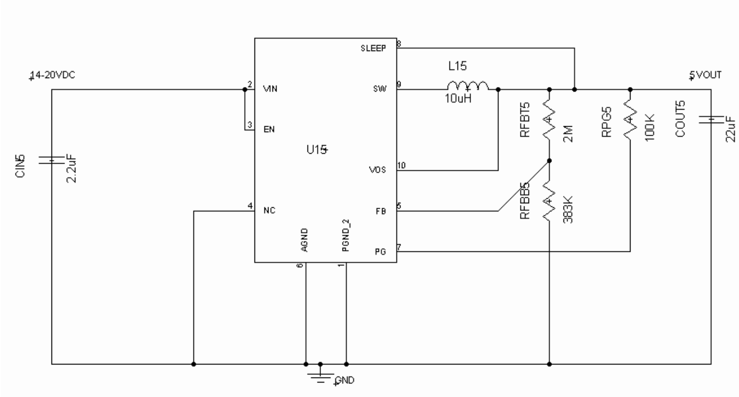


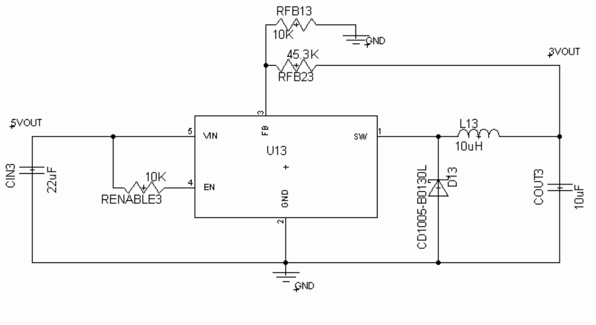
Voltage of Line	Current Draw
12V	450mA
5V	76mA
3.3V	198.5mA

Transformer

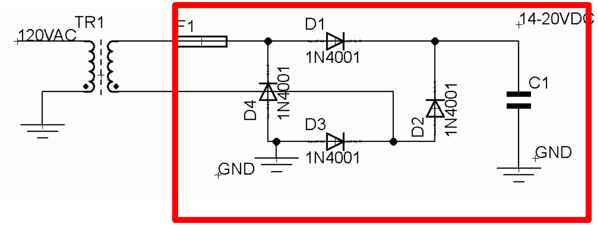


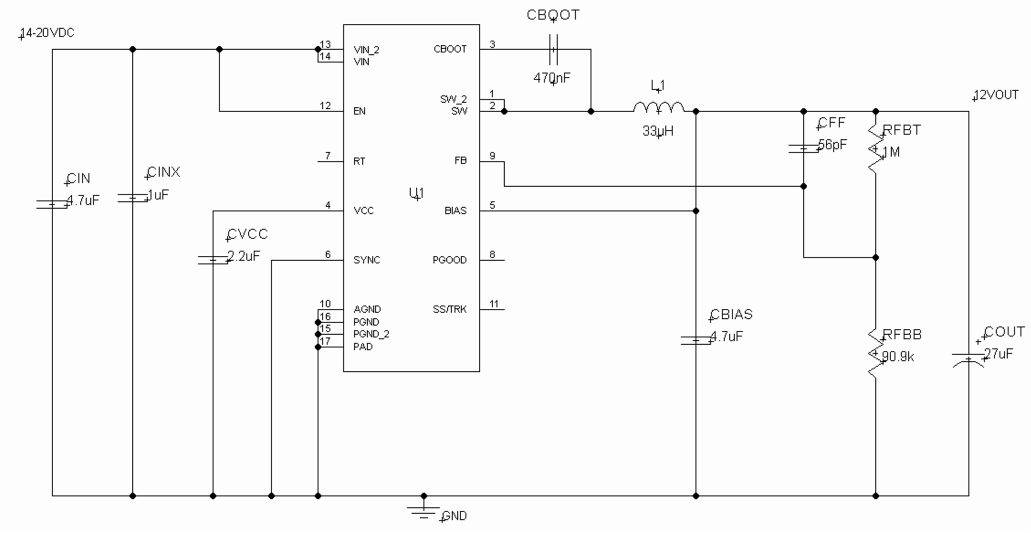


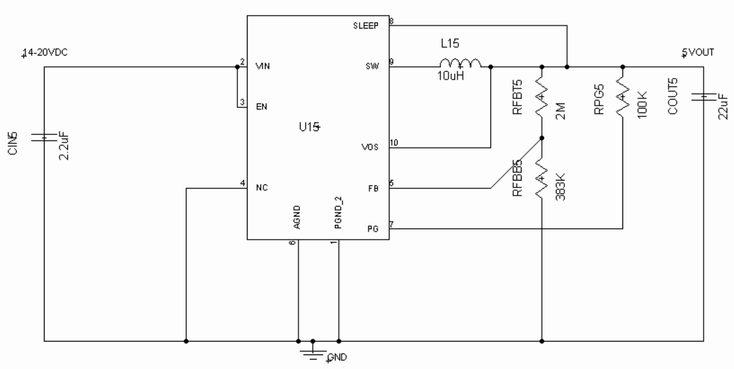


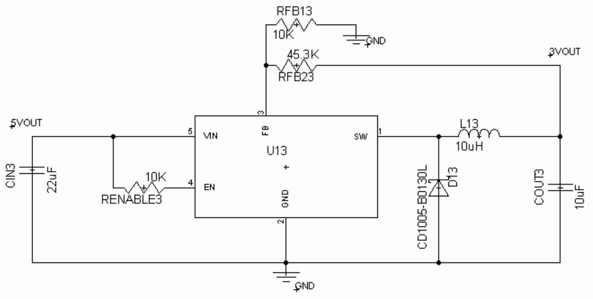


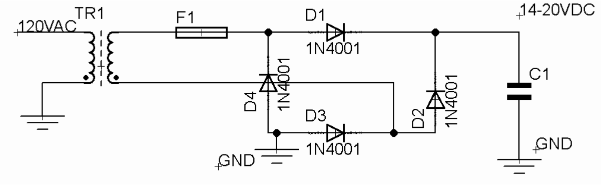
Rectifier

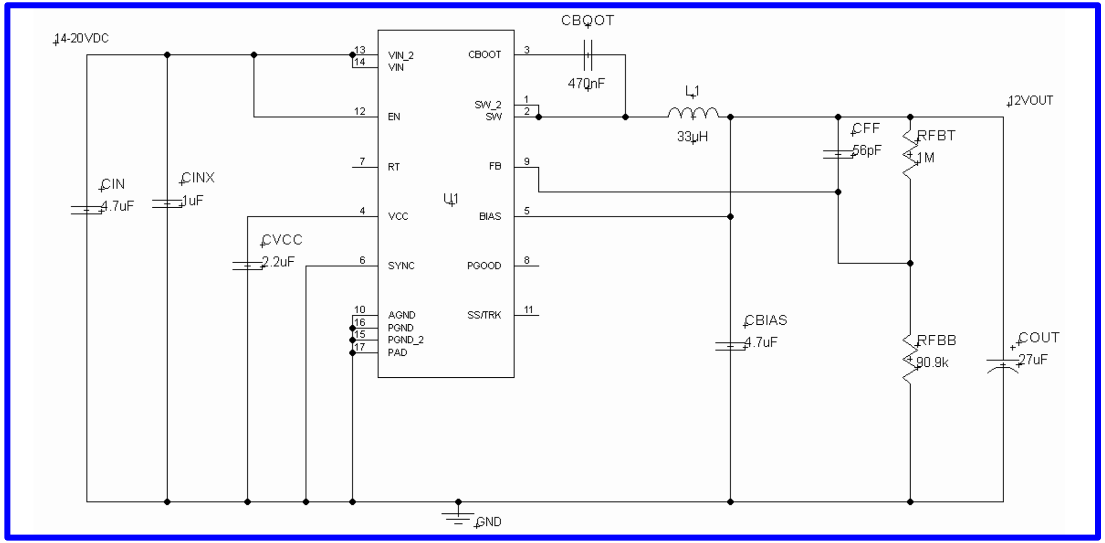

120VAC
TR1
F1
D1
1N4001
14-20VDC
D4
1N4001
D3
1N4001
D2
1N4001
C1
GND
GND


14-20VDC
CBOOT
470nF
L1
33µH
12VOUT
CIN
4.7uF
CINX
1uF
CVCC
2.2uF
U1
VIN_2
VIN
EN
RT
VCC
SYNC
AGND
PGND
PGND_2
PAD
CBOOT
SW_2
SW
FB
BIAS
PGOOD
SS/TRK
13
14
12
7
4
6
10
16
15
17
3
1
2
9
5
8
11
CFF
56pF
RFBT
1M
CBIAS
4.7uF
RFBB
90.9k
COUT
27uF
GND

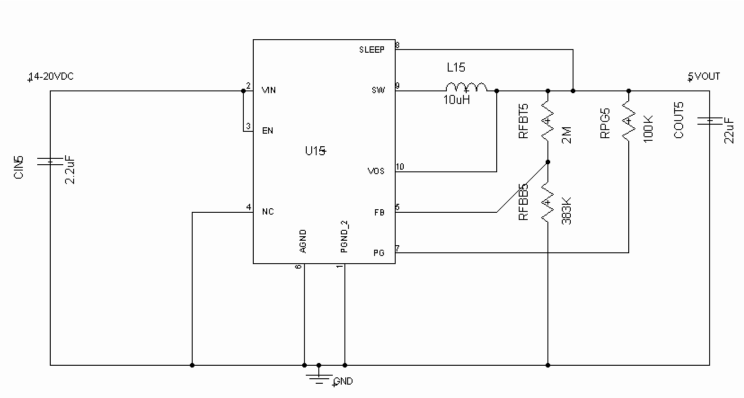

14-20VDC
CIN5
2.2uF
U15
VIN
EN
NC
AGND
PGND_2
SLEEP
SW
VOS
FB
PG
2
3
4
6
1
8
9
10
5
7
L15
10uH
5VOUT
RFBT5
2M
RFBB5
383K
RPG5
100K
COUT5
22uF
GND

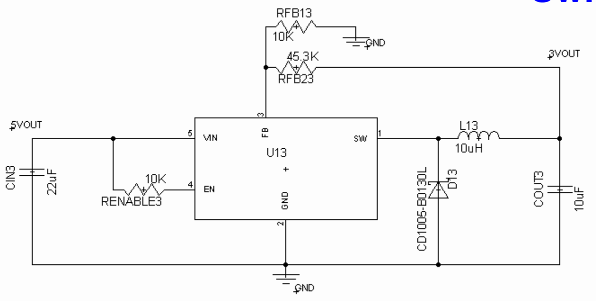

RFB13
10K
GND
45.3K
RFB23
3VOUT
5VOUT
CIN3
22uF
RENABLE3
10K
U13
VIN
EN
FB
GND
SW
5
4
3
2
1
L13
10uH
CD1005-B0130L
D13
COUT3
10uF
GND

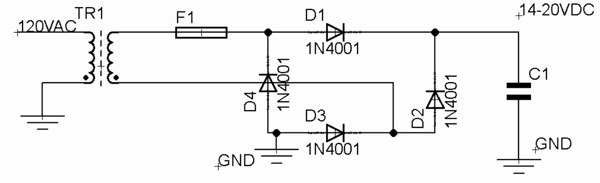


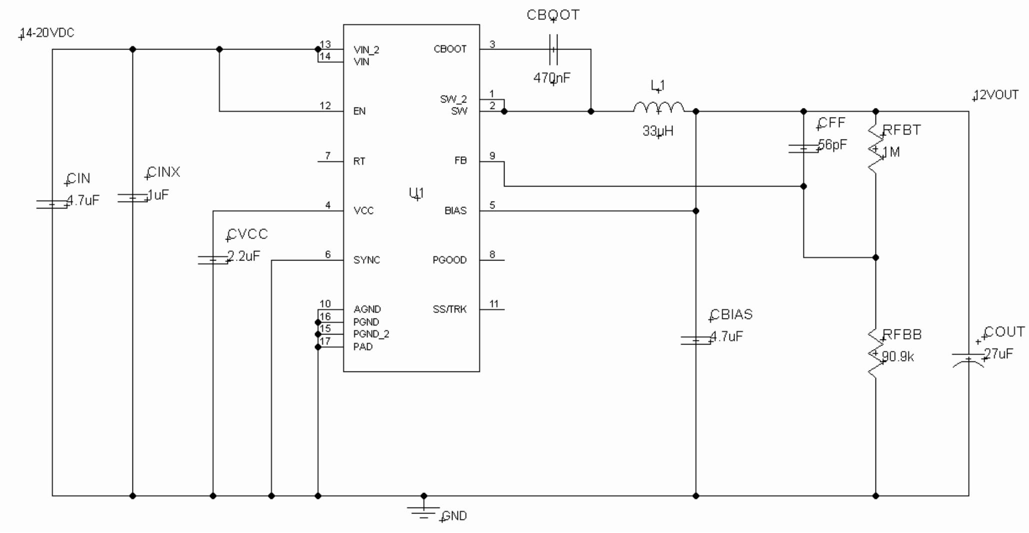


12 V
Switching Regulator

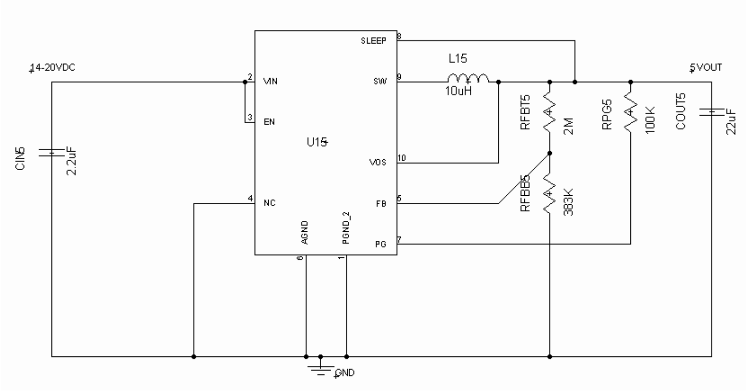


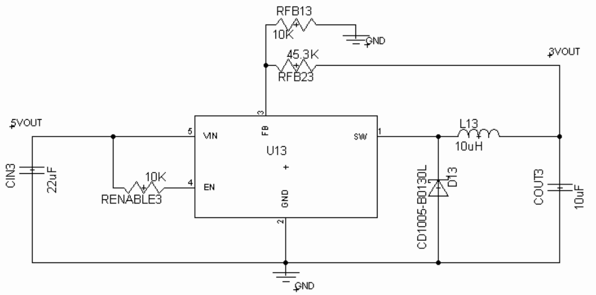


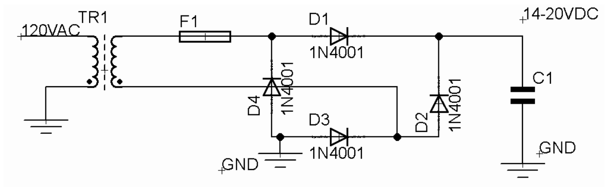


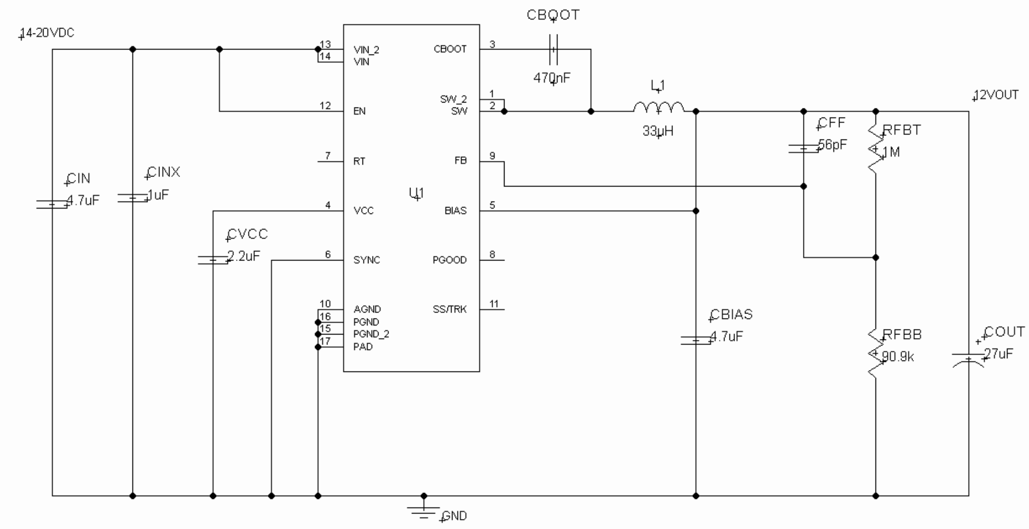


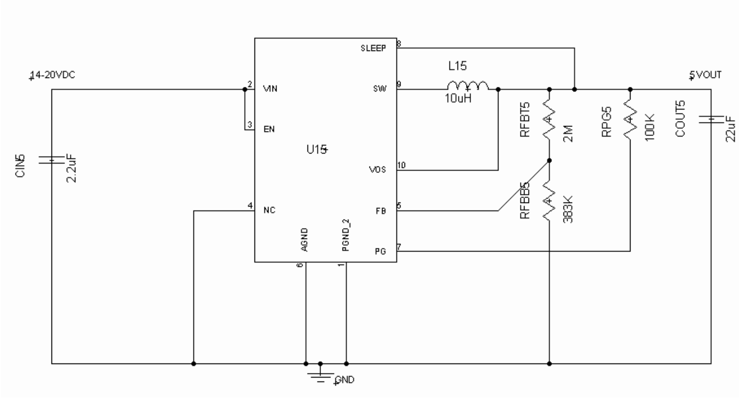
5 V
Switching Regulator



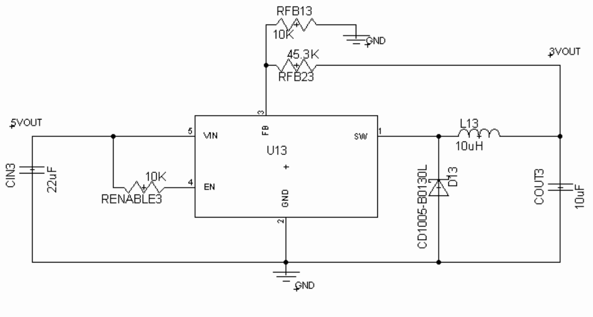








3.3 V Switching Regulator



Switching Regulators

- DC to DC conversions made with WEBENCH using current and voltage requirements

Regulator	Vin	Current Drawn	Efficiency
12V	18.4V	450mA	94.5%
5V	18.4V	211mA	85%
3.3V	5V	199mA	88.7%

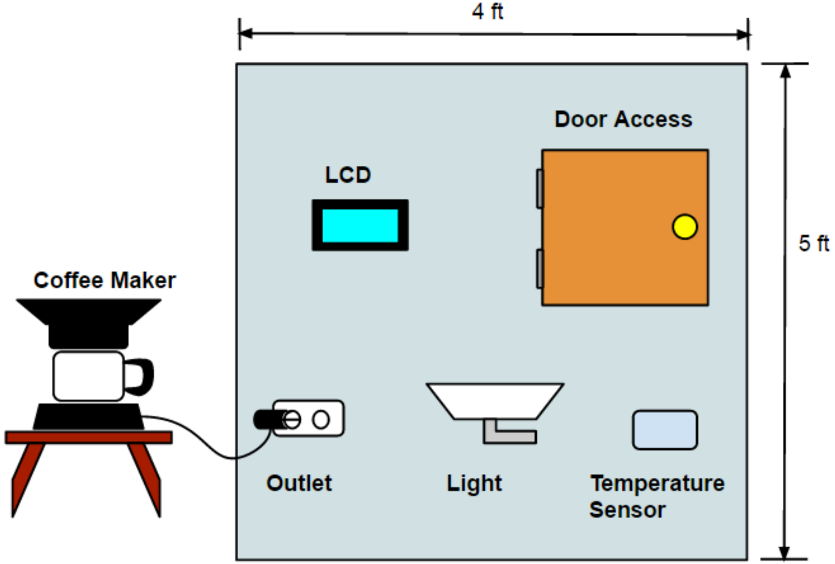
Power Board Layout

- Being designed in Eagle
- Switching regulators designs made with WEBENCH Designer and Power Architect
- Used existing footprints when available from Sparkfun and Ti library
- Used trace width calculator to ensure that trace widths were safe for the amperage allowable in the AC line
- Chose 0805 footprint size as a minimum for ease of soldering in prototype phase
- Order soon to be placed at OSH park for fabrication

WHCS

Final Prototype (Proto-Panel)

- The Proto-Panel will display the 4 control modules and the base station LCD on a single display board
- Each subsystem will be installed into the board in the same way that they would be in a home
- Board will use full scale components with the exception of a smaller door size



Budget and Financing

Item	Qty.	Subtotal	Item	Qty.	Subtotal	Item	Qty.	Subtotal
HC-05	1	$7.95	Electronic Strike	1	$21.94	3.3V Regulator	5	$4.85
NRF24L01+	6	$5.85	2.8” TFT LCD	1	$30.00	Rectifier Diode	20	$2.20
ATmega32A	1	$6.15	Misc. Capacitors	5	$5.80	Smoothing Cap	5	$8.80
ATmega328P	5	$18.50	Misc. Resistors	5	$13.35	120VAC Transformer	5	$56.20
AC Solid State Relay	2	$11.96	Terminals	12	$7.08	Fuse	5	$3.60
DC Solid State Relay	1	$1.71	Pocket AVR Prog.	1	$15.00	PCB Manufacturing		$144
16 MHz Crystal	5	$3.15	Pwr Board Passives	5	$25.32	Shipping		$30
3.3 Voltage Regulator (LDO)	5	$3.10	12V Regulator	1	$4.15			
Temperature Sensor	1	$1.50	5V Regulator	5	$10.50	**TOTAL**		**$442.66**

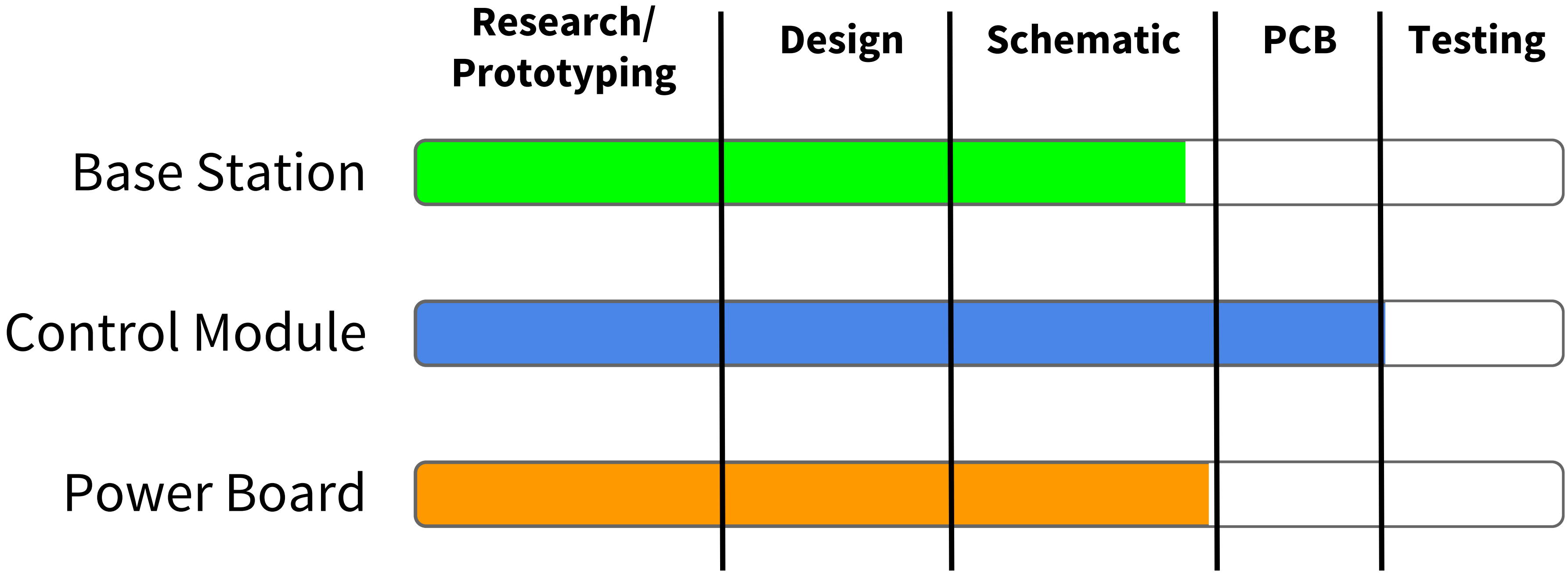
Sponsored by Boeing for $434.42

WHCS

Subsystem Progress

Key

🟩 - Grant
🟦 - Jimmy
🟧 - Joseph



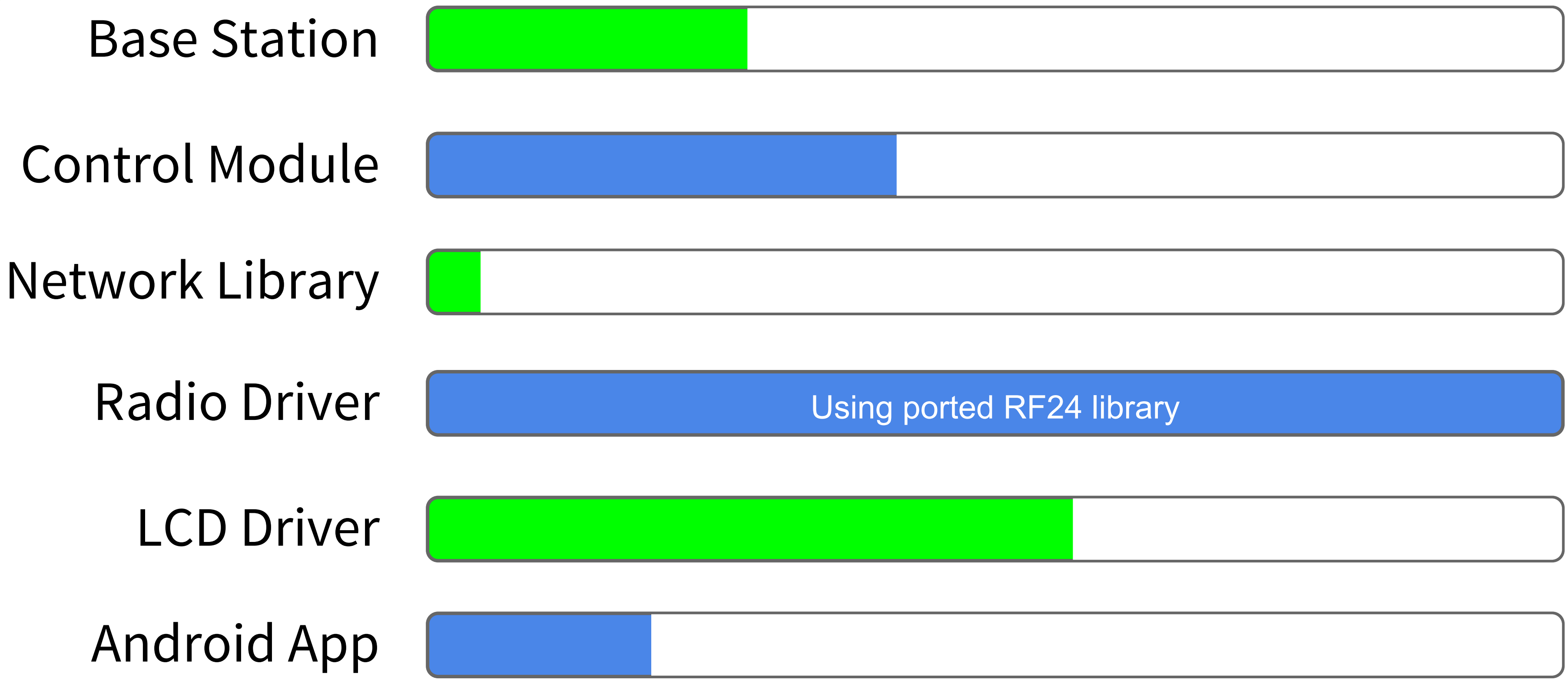
Code Progress

Key

🟩 - Grant

🟦 - Jimmy

🟧 - Joseph



Using ported RF24 library

Concerns

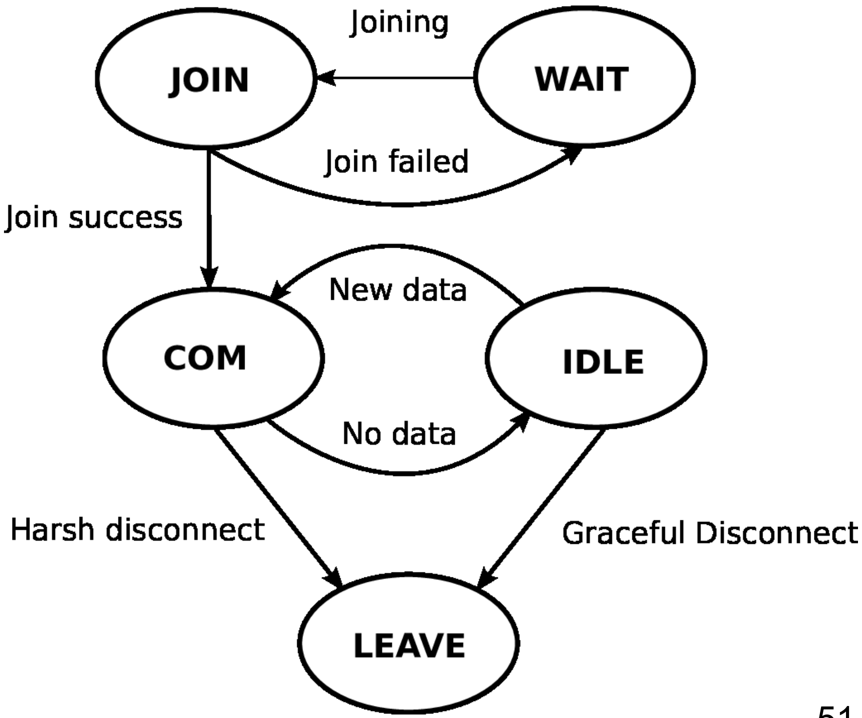
- Battery backup
    - **Possible solution**: stick with wall power
- AC heat dissipation (15A max) or heatsinking
    - **Possible solution**: thick AC traces and well rated terminals and relays
- 3.3V LDO on each board versus just power board
    - **Reasoning**: we want to decouple reliance on the power boards and still be able to program using 5V from USB
    - **Possible Solution**: only populate the 3.3V LDO footprint during development
- Base station processing power: how do we prevent radio starvation?
    - Has to process LCD, NRF, and BlueTooth simultaneously
    - Graphics drawing is expensive and blocks other tasks
    - **Possible solution**: interrupts or RTOS. May not actually be an issue

Wireless Home Control System

Questions?

Base Station and Control Module Interaction

- Control Modules will associate with the Base Station over the NRF
- The Base Station will maintain state for each CM
- During the initial handshake the CM will announce its capabilities to the BS for display purposes and packet decoding/encoding


JOIN
Joining
WAIT
Join failed
Join success
New data
COM
IDLE
No data
Harsh disconnect
Graceful Disconnect
LEAVE