Auto Logger

Vehicular Data Logging Interface

Hassan Siddiqui, Zachary Ross, Justin Wright

Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — The AutoLogger is an autonomous logging interface for both electric and internal combustion engine vehicles. The device will collect characteristic transportation data which can be utilized by the project sponsor, The Florida Solar Energy Center, to help improve the UCF campus parking infrastructure and raise awareness of the benefits of electric vehicles.**

***Index Terms* — Automotive applications, Electric Vehicles Global Positioning System**

I. Introduction

The Auto-Logger is an autonomous logging device which will collect a robust amount of information in order to meet the University's and the FSEC's needs. In order to adequately portray the advantages of electric vehicles, data on fuel combustion rates, engine stress, and oil levels will be collected on Internal Combustion Engine (ICE) vehicles. Additionally, the Auto-Logger has the capability to collect information on battery health, charge capacity, atmospheric temperature and many other parameters when interfacing with electric vehicles. With this information the FSEC can make a strong comparison between combustion engines and electric or even hybrid engines and their impact on both your wallet and the environment. To address parking infrastructure on the UCF campus the Auto-Logger will log information pertaining to the vehicle's location with an integrated GPS module, overall time spent on campus including time parked using the onboard clock associated with our chosen microcontroller, and even duration of time spent accessing one of the EV charge stations if integrated onto electric vehicles.

The Auto-Logger will incorporate customized microcontroller design and fabrication in order to minimize the housing space required within the vehicle. We expect the Auto-Logger to reside in a small and discreet package, enough to fit within the vehicle without imposing on the driver's legroom. This microcontroller will interface directly with the vehicle through the standardized OBD-II interface port, which resides in the cabin of the vehicle typically below the steering wheel. The AutoLogger will automatically upload all collected data to a MySQL database hosted by the FSEC in Cocoa Beach. To drastically reduce the operational costs of the Auto-Logger, Group 13 will make use of a Wi-Fi network to upload this data, as opposed to charging additional fees to the drivers or the budget for any cellular data used.

II. OBD-II Interface

Many different manufacturers use their OBD-II port in different manners. Older Japanese cars used ISO lines while American manufacturers used PWM and VPW, and newer cars use CAN. On top of this variation, many manufacturers use the open pins for proprietary uses (e.g. Alfa Romeo OBD command to check supercharger performance). The Auto-Logger will ignore the pins that aren't mandated by SAE J1962 because our device wouldn't be able to interface with them without potentially damaging the vehicle's computer or itself. In order to communicate to the OBD of the vehicle, the data transmitted and received from the car should be sent over an RS232 converter cable. Only 9 pins from the OBD-II port is needed in order to call the SAE J1962 required OBD-II readings. So in order to minimize the thickness of the cable and the space required on the PCB, RS232 DB9 type cable was chosen. OBD-II PIDs are hexadecimal based codes that are used to fetch data from a vehicle. The Auto-Logger incorporates only the PID commands that are outlined and standardized within SAE J/1939. The PIDs below outline the types of commands that will be implemented into the design. The modes that will be utilized along with their respective hexadecimal PID commands are as follows:

Mode 1 allows real time data acquisition from the time at which the PID command was sent from the Auto-Logger. PID commands 00, 1C, 20, 40, and 60, will allow the device to identify which PIDs and standards, if any, are available to extract data from and will determine which commands will be valid with the vehicle currently being data mined. PIDs 04, 0C, 0D, 10, 50, and 5E are all used to determine internal combustion engine load on the vehicle. Under typical conditions, the device will only need PID 04 to read the engine control unit's (ECU) internally calculated value. However if the vehicle doesn't support PID 04, then the Auto-Logger will have to calculate the engine load by retrieving all the variables manually from PIDs 0C, 0D, 10, 50, 5E and using the same engine load formula that PID 04 would utilize. PIDs 0C, 0D, 11, 1F, 5A, and 7F will be used to determine whether the car is currently being used by the driver. By using these metrics the Auto-Logger can be intelligent enough to determine if the driver had left the vehicle off and parked. This way the Auto-Logger can put itself into low power mode and keep itself from drawing too much power from the car battery. These readings such as vehicle speed can also aid in determining distance traveled and serve as a backup in the case that the GPS module has lost connection.

Mode 2 takes a snapshot of the current state of the vehicle. The hex PID commands are identical to the Mode 1 PID commands listed about, with the exception that the data extrapolated from this mode is given from when the freeze frame was created. The freeze frame is initiated once the Auto-Logger puts the OBD-II into Mode 2.

Mode 9 commands are useful to identify the vehicle the Auto-Logger is operating on. This will be useful when trying to sort data from multiple vehicles using duplicate Auto-Loggers in the future in a qualitative manner. A vehicle identification number (VIN) is a 17 - character affixed to every automobile since 1981. A VIN is the most reliable way to track what kind of vehicle is being operated on because no two cars built within 30 years of each other can share the same VIN. This will allow FSEC greater flexibility in organizing the data retrieved from multiple Auto-Loggers in the future.

Modes 3, 4, and 5 are irrelevant to project design specifications and do not aid in accomplishing any of the data procurement from the vehicle. These modes are only used when trying to diagnose "check engine light" errors by checking and clearing error codes from the vehicle's dashboard display. Because this device will not be used as a typical car-scanner, these modes and any data from them will be ignored by the device.

III. Power usage

The Auto-Logger will be powered solely from the OBD-II supplied 12V pin. Many devices such as the Wi-Fi module, GPS receiver, and SD card circuits require 3.3V - 5 V to operate in nominal conditions. In order to reduce the supplied voltage to a sufficient level without requiring the use of a heat sink sufficient spacing must be introduced to accommodate for a significant temperature gradient.

A standard lead-acid car battery contains a charge capacity of about 45 Amp-hours. That means that it could supply over two amps for 20 hours. A battery should not be discharged at a higher current draw, or asked to deliver more amps than its amp/hour rating divided by 10 in order to get maximum capacity out of it. During Low Power Mode (LPM) the MCU, Wi-Fi NIC, LCD, and GPS module only draw a total of 0.56mA. During peak current draw the device's components draw up to 0.575A. To ensure that the Auto-Logger will not be a liability to FSEC or any of the participants in their study, proper battery discharge rates on the vehicle's lead-acid battery are essential. Peukert's Formula for battery discharge was utilized to estimate the efficiency of the AutoLogger.

Peukert's Number is a characteristic measure of a batteries discharge rate and is determined through iterative hardware analysis performed by the battery manufacturer. But due to the many types of car batteries drivers may have installed onto their vehicle in a multitude of various conditions, the number can fluctuate between 1.1 and 1.3, the calculations used to design the AutoLogger use the worst case scenario of 1.3.

Generally batteries are measured in cycles or how many times they can be discharged and recharged before they fail to hold a complete charge. Different batteries provide different charging/discharging characteristics which influence a batteries life expectancy. The main issue at hand is that lead-acid batteries are not designed to be depleted lower than 80% of its charge capacity and will be damaged if dropped below that threshold. Thus the calculations below will prove that the Auto-Logger will not cause such issues even in the event that low power functionality is compromised.

$$\mathbf{20\%\ \textit{Discharge Time}} = \frac{\mathbf{45}}{\mathbf{0.575^{1.3}}} * (\mathbf{0.2}) = \mathbf{18.478}\ \boldsymbol{hours}$$

Equation 1 - Lead Acid Battery Discharge Time with Peak Load

$$\mathbf{20\%\ \textit{Discharge Time}} = \frac{\mathbf{45}}{\mathbf{0.00056^{1.3}}} * (\mathbf{0.2}) = \mathbf{151,914}\ \boldsymbol{hours}$$

Equation 2 - Lead Acid Battery Discharge Time with LPM Load

IV. Wireless Communications

The Auto-Logger will utilized the ESP8266 chipset on the ESP-07 Wi-Fi Module to provide the device with a reliable and power efficient internet interface. The ESP8266 chipset has a uniquely small package size with 32 individual pins which double as a microcontroller. While the module package will not take much room on the final board the team must plan to give additional space to accommodate the associated input pins, such as signal amplification antennas if desired.

It is important to note that the ESP-07 operates at 3.3V, a spike in current would likely cause damage to the module. While the designed PCB does have a 3.3V Voltage Regular on board, we must be aware of the device's capability in providing sufficient current to power this Wi-Fi module. This may be considered a design constraint, however insignificant it may seem.

The ESP-07 Wi-Fi module will require some additional hardware, such as capacitors for voltage regulation and resistors for current management, in order to properly communicate with the MCU and the FSEC server. While this module comes complete with a built in antenna port, we have put research into improving the capability of our device by reducing the required signal strength to establish a stable connection. In order to achieve this goal, an amplifying whip antenna is utilized to improve received signal strength. This is important as the Wi-Fi signal in the parking lots and throughout campus is patchy, which could result in a constant failure to establish a connection for some participants. However, precautions must be taken when selecting a specific device so as not to impose on the spatial necessities of the driver. ESP-07 comes equipped with a high frequency crystal oscillator, which can range from 26MHz to 52MHz, and a real time clock. Because the crystal oscillator can experience a large frequency drift based on its operating temperature, it is necessary to wait until the vehicle has reached a stable temperature to begin wireless communications. In order to meet this constraint, the Auto-Logger will only select and establish a wireless connection once the vehicle has stopped operation. This approach makes it possible for the device to remain asleep during rapid cooling of the cabin from the A/C, when the most significant frequency drift would be experienced. Additionally, the most stable connection will be established while the vehicle is stationary.

It is understood that the Auto-Logger, with a stable connection, will have the ability to dump its memory banks into the FSEC mySQL database within 15 minutes, while accommodating some miscommunications. Substituting values provided into the total transmission time would come to 13 minutes 19 seconds. Values were attained by assuming “worst case scenario” conditions for transfer. Data attained per day should not exceed 10 MB, accumulating 500 MB of data would take almost 50 days of continuous on-campus navigation. Since data will only be collected for vehicle operation while within campus bounds, the amount of data queued for transfer should never exceed 100MB. Moreover, a requirement for participants in this study is they should be driving to campus at least twice per week. This will allow the Auto-Logger to keep its data transfer down to a minimum, transmitting the small magnitude of data it has collected on a regular basis.

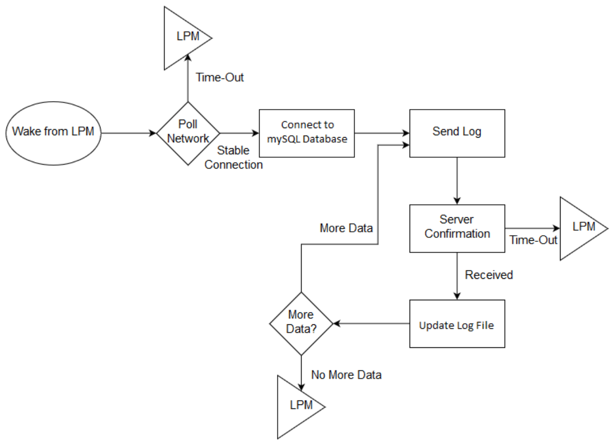
The Auto-Logger will perform all data transfer in a short period of time after the vehicle has been shut down while on the UCF campus. Even though the vehicle is shut down, and the alternator is off, the OBD-II interface will still provide 12V from the vehicle’s lead-acid battery.

While not in use the ESP-07 must be held in low power mode, specifically Light Sleep mode. The MCU will initialize the interface with the Wi-Fi module and immediately put it into Light-Sleep operation. While there is sufficient power to support the system even with the ESP-07 module running at all times, we believe that it is unnecessary for the device to operate on full power if not required. The Auto-Logger uses Light-Sleep operation, as opposed to the Deep-Sleep mode, because it supports an interrupt based wake-up support instead of an incremental wake-up service. Since the vehicles will be used for personal activities it would be undesirable for the Wi-Fi module to consistently wake up at predetermined intervals while the vehicle is nowhere near the transmission location.

In order to enter Light-Sleep mode, the ESP-07 requires a firmware update from SDK 1.3.0 to SDK 1.5.0, provided by the manufacturer Espressif. In order to flash the firmware update to the ESP8266 chipset, Group 13 utilized XXXXXX. Once the firmware has been updated, the typical AT command list is erased. As such, the ESP-07 will operate on code architecture created by Group 13.

Implementation

The Auto-Logger will implement the ESP-07 Wi-Fi Module in order to connect to a Wi-Fi network. The main MCU will initiate the state sequence once the GPS localization identifies the vehicle as “on UCF premise” and the vehicle’s engine shuts down. Once initiated by the MCU the Wi-Fi module will wake from Light-Sleep Mode and begin to poll the specified network SSID. With the confirmation of a stable network connection established, the communications state sequence will progress as depicted below:



Polling of the Wi-Fi network is critical for the establishment of a reliable connection with the FSEC Server. The received signal strength indicator (RSSI) is utilized to uphold a tight threshold on the signal strength. To ensure a reliable connection, the Auto-Logger will enforce a threshold of -80dBm. This threshold will guarantee accurate packet transfer and a drastic reduction in packet loss leading to repeated transmissions. RSSI is a measurement of how well a device can hear a signal from an access point or router. Not to be confused with the associated transmit power of the access point or router, the RSSI is a value useful for determining the strength of a specified signal you can detect. The ESP-07 will rate the specified signal on a scale from 0 to 255 where the higher the number the stronger the signal is.

When the vehicle engine is shutdown, the Auto-Logger believes this would be the most reliable time to connect to the UCF Wi-Fi, specifically because the device is stationary. When prompted by the GPS and OBD-II data streams, the MCU will wake the Wi-Fi module and initiate the connection sequence with the specified network. After a connection is establish the MCU will begin to poll the signal strength. Open-source libraries for the ESP-07 provide readily available RSSI interpretation for the ATMega2560, supplying the Auto-Logger with a clear and intuitive capability to study the signal strength and determine whether or not such a network could reliably support the transfer of our mission critical data.

The Auto-Logger will utilize secure Transmission Control Protocol (TCP) as its medium of communication which comes equipped with many features which provide reliable support. In order to avoid conflicts with the UCF Wi-Fi firewall the team will demo the project on a temporary Hotspot network.

As recommended and made readily apparent by innumerable implementations, Image or Binary Mode is the representation implemented in the Auto-Logger because reliability is key in such a system. This data representation will also reduce the complexity of the serial transfer between the MCU and the ESP-07 as well as the TCP communication between the client and server, as various conversion will not be required on both ends of the communication process. Various open-source libraries are available and will be implemented to perform these conversion on the fly

Communication with the FSEC server will require conjoined effort from the MCU, Wi-Fi module and the SD Card in order to gather, organize, prepare and transmit the desired data logs. Analysis by the server will be required to ensure the file was received successfully. Instead of deleting logs as soon as they have been transmitted to the server, the Auto-Logger will implement a circular storage methodology. This technique will continue to save data logs on a per second basis until the entire storage unit has been filled to capacity. Once storage capacity has been reached, the device will begin to overwrite files starting from the oldest file and progressing towards the newest thus completing the circle.

The Auto-Logger will connect to the FSEC webserver securely with the use of a private username and password combination. Communication will utilize TCP port 443 as it, along with TCP port 80, remains open even when such a server is stationed behind a firewall. Once a connection has been established with the server the Auto-Logger will transmit the contained data files, saving them into a MySQL database provided by the FSEC. The Auto-Logger, through the use of a data transfer log, will keep track of transferred data ensuring the server only receives what is needed.

V. GPS Localization

The Auto-Logger will utilize Adafruit's FGPMMOPA6H GPS module for accurate real-time vehicle localization. This module implements a MediaTek (MTK) chipset to provide features pertinent to the goals of the Auto-Logger such as data logging and acquisition rates which exceed 5 Hz. While the package for the device is very small, additional space on the PCB must be devoted in order to accommodate the modules associated hardware such as antenna ports.

The FGPMMOPA6H is an ultra-compact, patch on top, 66-channel GPS Engine board equipped with antenna module and MTK chipset. This receiver provides an RF solution that is high in position and speed accuracy performances, with high sensitivity and tracking capabilities in urban conditions which are ideal for the purpose of the Auto-Logger. This GPS module is an all-inclusive package providing the MKT chipset with a POT antenna, 32 MHz Crystal clock, 16 MHz temperature compensate crystal clock and main Low Dropout (LDO) Voltage Regulator.

The temperature compensate crystal clock (TXCO) is implemented in low-cost designs requiring a precision frequency source within a small space. This device has temperature compensation circuitry to suppress output frequency deviation caused by temperature changes in the surrounding environment. Such compensation is highly advantageous for the Auto-Logger since the device will remain in a vehicle for the entire product life-cycle. Atmospheric temperature in a vehicle can easily range from 25° Celsius to 70° Celsius. Experiencing the complete temperature range within an hour's time, up to several times per day is a result of typical vehicle usage, as such the TXCO is a significant asset to the reliability of the GPS acquisition system.

Temperature Compensated Crystal Clock oscillators with standard compensation techniques achieve fractional stabilities of around ±1 parts-per-million (ppm) for a temperature range of -40° C to 85° C.

Each individual TXCO requires personalized network compensation to be matched with its specific crystal. Subsequently, the cost of a TCXO is closely related to the difficulty of the frequency versus temperature specification. During operation, a voltage change causes a change in the capacitance of the varactor diode. This results in a change in the frequency of oscillation. The thermistor network is tailored to the crystal in order to force the voltage to vary with temperature such that the crystal's frequency variation characteristics versus its temperature change are adequately compensated for.

The LDO voltage regulator supplies the MTK chipset with performance capabilities optimized for battery-powered systems delivering low quiescent current, thus prolonging battery life. The LDO utilized in the FGPMMOPA6H GPS module is the Richteck RT9193 300mA, Ultra-Low Noise, Ultra-Fast CMOS LDO Regulator. This component provides the FGPMMOPA6H with a pristine power signal, regulated of all noise and jitter, fast switching speeds and ability to draw sufficient current even as battery voltage continues to deplete. Moreover, the LDO voltage regulator is driven with low-ESR ceramic capacitors thus reducing board space required to perform the same job. The RT9193 LDO voltage regulator helps the FPGMMOPA6H GPS module achieve such a level of efficiency and reliability, critical for low-power mobile applications.

A geographic coordinate system enables every location on a spherical planet to be specified by a set of numbers or letters, or even symbols. Coordinates are often chosen such that one number represents vertical positioning while the other represents horizontal positioning. The standard depiction of global positioning uses longitude and latitude, elevation is sometime incorporated as additional information. Latitude depicts a point on the Earth's surface where the angle between the equatorial plane and the straight line that passes through that point and through (or close to) the center of the Earth. Latitude ranges from 0° at the Equator to 90° at the North or South Pole. Longitude depicts a point a point on the Earth's surface is the angle east or west from a reference meridian to another meridian that passes through that point.

Latitudinal and longitudinal information are usually communicated using decimal or Degree/Minute/Second notations. The GPS module implements a proprietary Degree/Minute/Second notation regulated by the NMEA standard - 0183 version 2.

Due to NMEA restriction, GPS information clearly has an inherent error rate associated with it when compared to the typical coordinate system notations, specifically Degree/minute/second. The NMEA regulated notation sacrifices the high resolution associated with the seconds metric of the typical notation for a system tuned towards the abilities of the technologies implemented. In order to illustrate a greater localized accuracy the standard sacrifices localization resolution, the regulated notation removes the seconds unit, instead implementing a decimal into the minute unit. Due to the introduction of a decimal unit with 6 fields of accuracy the error associated with ignoring the seconds unit is bypassed, securing the error at the typical rate of acquiring a reading within 3 meters of its actual location 95% of the time.

In order to determine whether or not the participating vehicle is within campus bounds, the MCU must take the longitude and latitude provided in by the FPGMM GPS module and compare it to the designated campus footprint. The design team is working with the University of Central Florida to acquire realistic campus boundaries to maximize data accuracy. In the event that UCF is not permitted to release such information, the design team can acquire accurate coordinates through the United Stated Geological Services (USGS). The USGS provides free and highly accurate, up to 1 arc-sec, elevation and terrain data for the entire United States of America. Through the use of their applied resources, the design team can attain an accurate footprint of the UCF campus and possibly its sister locations.

VI. Microcontroller

The Atmel ATmega2560 is the chosen MCU utilized by the Auto-Logger. It will interface with every single module embedded within the device PCB. The ATmega2560 does not come pre boot loaded with any firmware and will have to be manually programmed through the use of an AVR programming software or with another boot loaded ATmega2560. This process is required in order to upload the Auto-Logger software onto the MCU

The ATmega2560, has 54 digital input/output ports, 15 of which can support PWM outputs. The main reason this board was preferred is because it has the capability to support 4 UART hardware serial ports. Because the ATmega2560 needs to interface with all the major modules of the Auto-Logger, it must be able to accommodate the use of several serial lines in parallel. It has 7 times more flash memory available in the chip which provides the ability to write a larger program and have more available RAM at the user's disposal if the capability is needed.

The ATmega2560 is programmed using the Arduino Integrated Development Environment (IDE) or its predecessor, Processing. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension (.ino). Arduino IDE will be used to debug and implement the final software and is readily available for free because of its open source nature.

The ATmega2560 is only available in surface mount 100 pin Thin Quad Flat Pack (TQFP). BGA packaging is very compact and allows for a very small footprint on the manufactured PCB, unfortunately it will be a very cumbersome feat to attempt to troubleshoot any BGA package due to all the conductive contacts being hidden under the IC. It would also be a challenging undertaking to assemble the PCB even if there is not a need to troubleshoot the device. The TQFP package of the ATmega2560 will allow us to easily troubleshoot the IC because all the pins are exposed and easily probed. The team is also experienced enough in soldering to assemble the PCB while using TQFP components. Additionally, by using a TQFP packaged IC the need for multilayered PCBs will diminish because the leads can be easily managed on the surface layer.

VII. Temperature Sensor

The TMP36 sensor is a low voltage, precision centigrade temperature sensor. This sensor provides an output voltage which is linearly proportional to the centigrade temperature scale. The TMP36 can provide typical accuracies of ±2˚C over the -40˚C to +125˚C temperature range and does not require initial external calibration to deliver such accuracies.

The low output impedance of the TMP36 simply the interfacing process and is intended for single supply operation from 2.7V to 5.5V, which lays along the range the Auto-Logger will supply to its other components. This model of temperature sensor produces 125mV output at 25°C and scales linearly as temperature increases or decreases by a factor of 10mV/°C.

The TMP36 temperature sensor, consisting of three pins, will interface directly with an analog pin on the ATMega2560. Using the two simple conversions previously stated, the MCU will be able to attain the ambient temperature of the vehicle cabin to within 1˚C. The value produced by the TMP36 ranges from 0 to 1023 based on the temperature it is experiences. With the value stored as a voltage measurement, the MCU can now convert it directly into a temperature measured in ˚C.

VIII. Real Time Clock

The DS1307 real time clock will be used in low power mode when the car is off. The DS1307 serves to keep track of how long the vehicle is parked in one location until the vehicle is turned on and begins moving again. The DS1307 will use backup battery power while the device is in low power mode. The DS1307 uses 8 bits and 8 address lines to keep track of time from seconds to the year. The day of the week register increments at midnight and the days are defined by the user, such as 1 equals Sunday. The accuracy of the clock is determined by the 32.786 kHz crystal oscillator. When reading and writing the time and date registers, secondary buffers are used to prevent error when the internal registers update. When the time and date is read the buffers are synchronized to the internal registers. The time information is read from the secondary registers while the clock continues to run. This would eliminate the need to re-read the registers in case the internal registers update during a read.

The DS1307 operates as a slave that needs to be controlled by a master device that generates the serial clock, controls the bus access, and generates the start and stop conditions. The data transfer can only be initiated only when the bus is not busy. While data is being transferred the data line must remain stable whenever the clock line is high. Changes in any data line while the clock line is high will be interpreted as control signals.

To start a data transfer change the data line from high to low while the clock is high. To stop a data transfer, change the data line from low to high while the clock is high. The master device must acknowledge the reception of each byte by generating an extra clock pulse with the acknowledge bit. A device that acknowledges must pull down the SDA line during the acknowledge clock pulse so that the SDA line is stable low during the high period of the acknowledge-related clock pulse. On the last byte a master must signal an end of data. To do this, the slave must leave the data line high to enable the master to generate the stop condition. To run without power from the car the DS1307 will use a CR2032 battery to operate. The CR2032 is a small lithium coin cell battery that has sufficient voltage output of 3V to operate the DS1307. The DS1307 is needed because when the ATMega2560 goes into low power mode the clock shuts off failing to track the time that has passed while in low power mode.

IX. Local and Remote Storage

The Data collected from the OBD-II port will be stored locally in a microSD card. The data needs to be stored locally because the vehicle will not always be able to immediately transmit data. The microSD card can vary in size depending on the interval between data transmissions to the FSEC servers. As stated previously, the microSD card is of sufficient size such that the available memory will not be depleted in the event that the vehicle is unable to transmit data on a regular basis.

The MM74HC 4050M hex inverting logic level down converter, LP2981-N low dropout regulator from Texas Instruments, and a SD/MMC card reader are the components used to read the microSD.

The microSD Card will be formatted using the FAT32 file system which is directly compatible with the Auto-Logger's ATMega2560 chipset. The 32 bit system is compatible with the microcontroller. Using the Arduino library we can format the microSD card and verify it is formatted correctly. The chip select pin should be connected to pin 53 so the value in the library should be set to 53. The microSD card will be inserted into a sketch to test if the volume type is FAT32. There are several functions for microSD Card memory management that can be used in the Arduino library when formatting the microSD card.

As recommended by the FSEC the Auto-Logger will call a web application and post the recently collected data to a server. This server will be implemented using a mySQL database used to store and manage the dataset received from the Auto-Logger. All communications will be secured, sent through TCP port 443, and limited to authenticated users only. The server will require the capability to ensure received files have not been corrupted or altered in any way. The MySQL database provided by FSEC will be created around these constraints and any additional concerns brought to the attention of either the FSEC faculty or the design team.

The Auto-Logger will act as a node application to save specified data files to the mySQL database in Cocoa Beach, FL. In order to efficiently communicate with this mySQL database, the Auto-Logger will excel at establishing a connection, pushing data, awaiting confirmation and managing the memory structure. In the event that some files have been corrupt on the server side, FSEC requests the Auto-Logger have the ability to keep track of all transferred data and only transmit what is needed by the server. Depending on the capabilities of the server the Auto-Logger may need to perform this action as a stand-alone device. The Auto-Logger should not continue to the next file to transmit until it has received confirmation from the server that the file was received and is in a functional condition.

Coding Architecture

The final software architecture will incorporate a combination of PHP and C++ to program the ATmega2560. PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Many of the server side architecture will be handled by FSEC employees and is not completely within the scope of the architecture outlined above. The software on the Auto-Logger will only use PHP as a conduit to relay information to the server in manner that this specific server can understand and interpret. Any PHP code in a requested file is executed by the PHP runtime, which is located on the server side application.

Oracle implements an open source database relational database management system (RDBMS) named MySQL. FSEC has specified that they only have Oracle servers that will be receiving the data collected by the Auto-Logger. Because of MySQL's open source nature, documentation on how to communicate and implement the device will not be a large barrier to overcome.

The overwhelming majority of the Auto-Logger design and program flow utilizes C++. The multitude of open source libraries that are advised to use with the ATmega2560, Wi-Fi Network Interface Card (NIC), real time clock (RTC), and the chips within the OBD-II communications board, are all written in a stable form using C++.

During first installation of the Auto-Logger it will retrieve data such as the VIN, identify what fuel type the vehicle uses, and which OBD-II standard communication protocol it uses to interface with the vehicle's electronic control unit (ECU) such as CAN, VPW, ISO, etc. This ensures the Auto-Logger does not use the wrong PID commands to retrieve the data from the vehicle.

The code-flow of the Auto-Logger attempts to activate Low Power Mode (LPM) whenever possible to decrease current draw from the vehicle's battery supply. By minimizing power draw at opportune moments, it will diminish thermal buildup in the separate ICs and power regulators on the PCB. This will increase the longevity of the hardware and allow FSEC to use the device for longer periods subsequently gathering more data for their study.

The liquid crystal display (LCD) will be used to display to the driver what the Auto-Logger is doing. It will notify the driver when it starts logging data from the car, alert the driver when it had started saving his or her vehicle's location to the microSD. This way the driver knows when vehicle is being polled. The LCD can also be built upon in future iterations of the Auto-Logger to display useful vehicle information to driver or to notify him or her if the ECU detected a fault. Many of those features fall outside of the scope of the current design effort for this iteration but will be detrimental for future developers of this device if an LCD isn't ever incorporated into the initial design iteration. This feature also will be useful for the driver so that they are knowledgeable of any action taken while driving their car.

The Auto-Logger must ensure that the vehicle is on campus before any data logging is performed. The GPS will perform intermittent pings to check if the vehicle has entered UCF campus grounds, data retrieval will begin to accurately and immediately store the data stream as soon as the vehicle crosses the campus boundary. This also shows that the device is indeed tracking the study participant's vehicle location at all times when the vehicle is in operation, but the privacy of the driver is kept intact because this location data isn't saved, in any form or fashion, on the FSEC database nor on the device's internal memory. The GPS location will be intermittently saved on temporary variables within the MCU to be compared with the predetermined campus bounds, saved within the MCU memory, to evaluate whether or not the participant's vehicle resides within the UCF campus.

Once the Auto-Logger has determined that the vehicle has entered campus grounds, it will enable the microSD and OBD-II communication modules and prepare them to begin fetching data from the ECU on a second-by-second basis, as requested by the FSEC sponsors. The speed of the ECU on most vehicles permits the retrieval of 100 readings from the OBD-II port, through a serial to UART conversion interface, every second. A typical microSD card has a read and write data throughput of 4-6 MB/s which will easily save a single data point per second, as the typical cards have an average throughput of 4,050 KB/s which is more than enough to log multiple data points to a text file in one second.

The preferred method to track whether or not the vehicle is in use or not is by procuring the many available metrics from the ECU through the use of the OBD-II hexadecimal PID commands. These commands are all being utilized from Mode 1 PIDs so that the MCU can identify exactly when the vehicle is not in use with almost negligible time delay. PIDs 0C, 0D, 11, 1F, 5A, and 7F will be used to determine whether the car is currently being used by the driver. By using these metrics the Auto-Logger can be intelligent enough to determine if the driver had left the vehicle off and parked. This way the Auto-Logger can put itself into low power mode and keep itself from drawing too much power from the car battery. These readings such as vehicle speed can also aid in determining distance traveled and serve as a backup in the case that the GPS module has lost connection.

XI. Conclusion

This project has proven to be a significant and valuable experience for each person in Senior Design group 13, especially when accounting for the lack of project experience appointed through the engineering curriculum. We have applied the concepts introduced in our classes and ideas developed on our own to bring our project into reality. Additionally, the group gained critical experience in management and professionalism through the support of and interaction with the Florida Solar Energy Center, the project sponsor.

In the end, this design experience has taught us a lot about ourselves, including communication skills and relative engineering skillsets, as well as aspirations for the future. With the completion of the Auto Logger design and fabrication and Senior Design 2, Group 13 will hand-off the final product as well as all materials to the sponsor, the Florida Solar Energy Center.

Acknowledgement

The authors wish to acknowledge the assistance and support of those at the Florida Solar Energy Center for providing us with financial support and an honorable design goal. Additional thanks go to our peers, mentors and colleagues for sharing this strenuous experience with us.

Biography

Zachary Ross, a 22 year old electrical engineering senior at the University of Central Florida. He is presently employed at UCF in the College of Sciences Physics department assisting in the design and development of the QPACE CubeSat. He hopes to one day work alongside NASA engineers and contractors in the pursuit of greater scientific discoveries.

Hassan Siddiqui is an undergraduate working towards a bachelor's degree in electrical engineering at University of Central Florida. He is currently an intern at Lockheed Martin MFC working in Advanced Manufacturing Technologies.

Justin Wright, a 22 year old electrical engineering senior at the University of Central Florida. He hopes to pursue a career working with MEMs devices after completing his degree.

References

[1] "Arduino - Tutorials." *Arduino - Tutorials*. N.p., n.d. Web. 01 April 2016.

[2] "ESP8266 Community Forum" *Everything ESP8266*. N.p., n.d. Web. 05 June 2016.

[3] "OBD Modes and PIDs" *OBD Modes and Configuration (PID)*. N.p., n.d. Web. 07 March 2016

[4] W.E. Jones "Notes on Batteries" <http://www.gizmology.net/batteries.htm> 2003