**Automated**

**Plantation System**

* **Harshil Sheth**
* **Rahul Yamasani**

**ECEN 5023 – Project Report**

**Executive Summary:**

This project aims at introducing automation in the plant growth process to ensure a healthier lifecycle for plants. According to a US survey, about 63% of home-grown plants die because of the insufficient care taken by their owner. This project could prove to be a boon for people who like growing their own plants but fail to do so because of a busy schedule.

The fundamental idea is to monitor several environmental parameters affecting plant growth and alert the user asking him to ensure the most suitable environment for plant growth. These parameters include temperature, ambient light, soil moisture level and water level in the tank. These values are notified to the user using an EFM32 Leopard Gecko as a sensor node transmitting data to the user via an ATSAMB11 BLE. We also allow the user to give back commands to control parameters like fan speed using his phone.

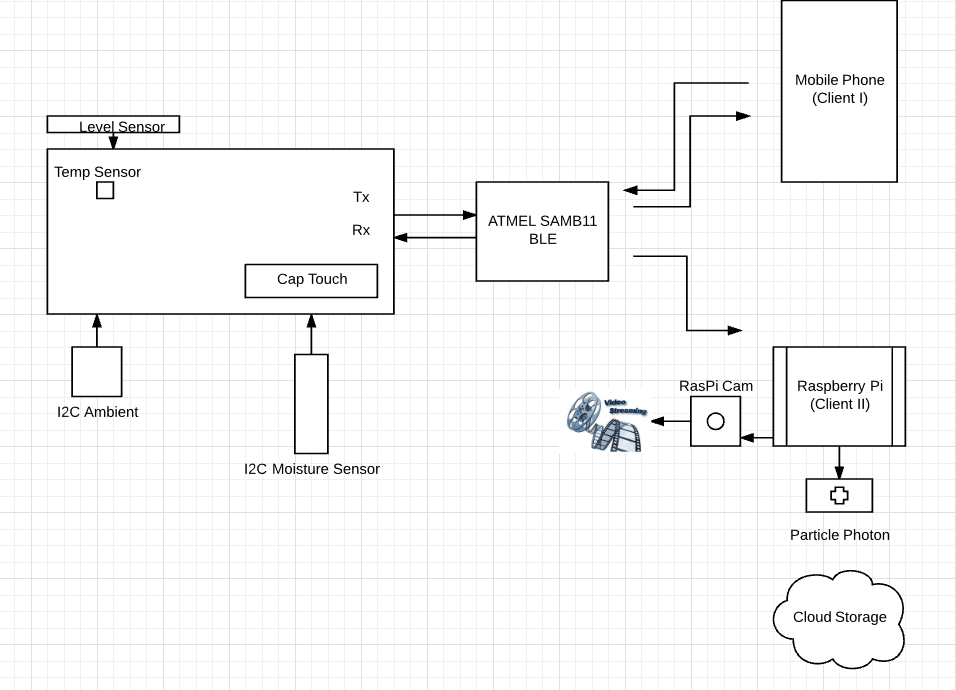
To eliminate the dependency on Bluetooth (limited range), we use raspberry pi as a sensor hub which takes data from the BLE and logs it onto the cloud using a particle photon board. This hub could be a host to several other nodes for future expansion. We have also interfaced a camera to the raspberry pi to capture real time images of the plant and allow live video streaming of the plant from any remote location.

This report provides a detailed description of the project and the test cases implemented to ensure robustness of the product. It also discusses the issues that we faced throughout the course of the project.

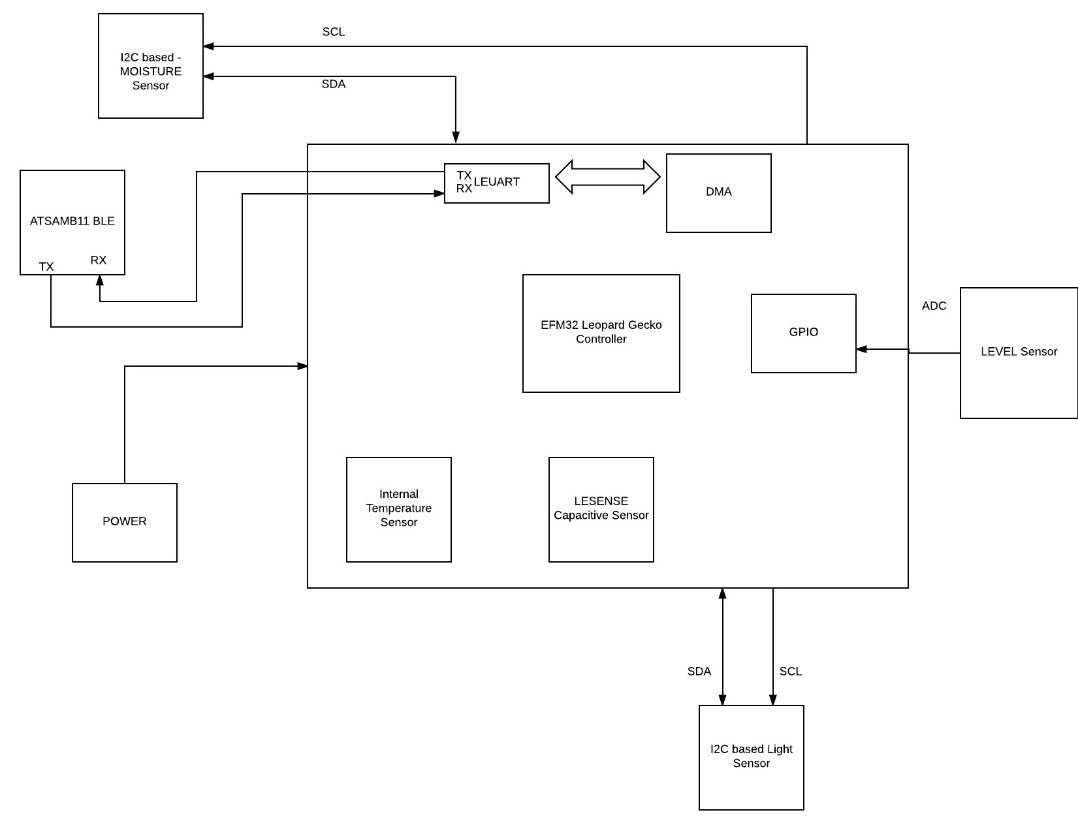
**Introduction:**

The idea of this product came up because of a belief that the process of growing your own plant in your backyard is similar to looking after a new born baby. Both require attention and care during their initial growing days. The only difference between them is that a baby can alert you (by crying) when it needs food but a plant cannot. This is where our product steps in. The system tries to maintain the best possible environment for a healthy plant growth leaving very little dependency on the user. It also prompts the user to take necessary actions by alerting them on their phone. Thus, this product is an attempt to ensure that no plant dies because of lack of care.

**Hardware Block Diagram:**



**Figure 1: Overall Block Diagram of our project implantation**

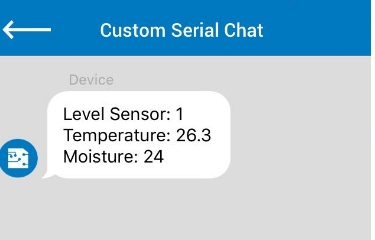
****

**Figure 2: Leopard Gecko Interface Block Diagram**

**Description (Step-by-Step Functionality):**

**Leopard Gecko and ATSAMB11 BLE:**

* The user activates the system by swiping left to right on the LESENSE Touch Sensor present on the Leopard Gecko.
* To save power, the sensor node takes a reading every 5 minutes using an RTC as system tick.
* The moisture sensor communicates with the Gecko using I2C communication. It gives out the moisture reading in the soil. The average range varies from 650 for dry soil pods and about 900 for an absolutely wet soil pod.
* Ambient light is measure by a TSL2561 Light to digital converter which is also an I2C based sensor. Depending on what the light reading is, the onboard LED light intensity is varied between for brightness levels using PWM. In real, we intend to use a 12V LED strip to provide sufficient light to the plants.
* Room temperature is measured by an onboard temperature sensor.
* We also intend to have a water outlet tank connected to the product. A level sensor will measure the amount of water (in inches) in the tank and alert the user of the water level.
* All the sensor data is transferred to a BLE via a circular buffer using serial communication.
* The BLE provides a Bluetooth connection medium between the Leopard Gecko board and the user. The BLE uses two profiles, the chat profile and the temperature profile to display all the sensor data to the user.
* In the chat profile, the user receives a message every 5 minutes that has all the sensor readings which looks something like this:



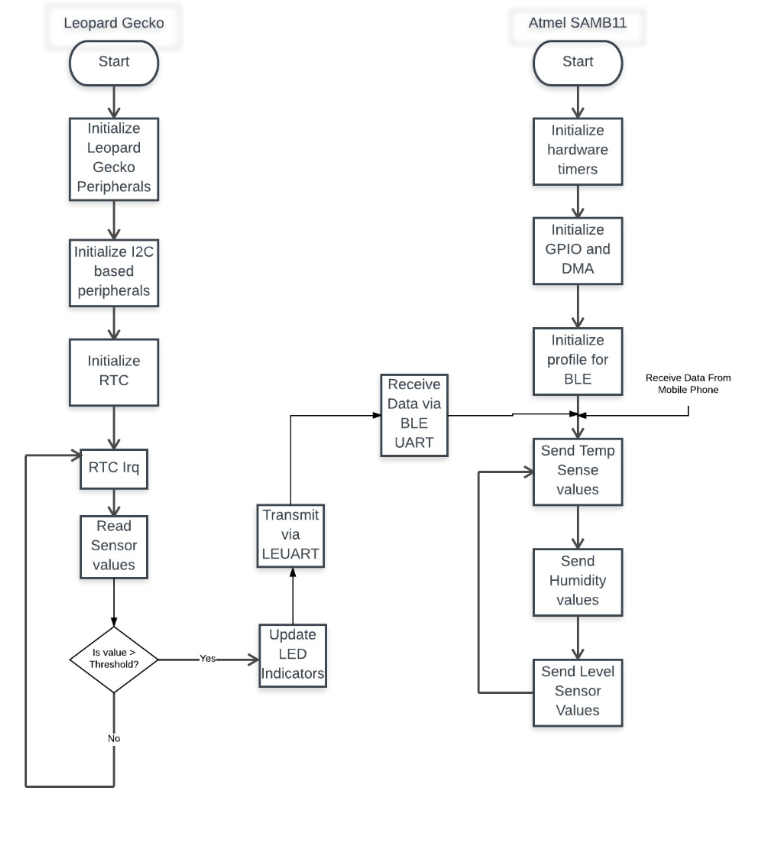
**Figure 3: Chat Profile Display**

* The user can disconnect by writing a ‘Stop’ message.
* If the temperature goes beyond a threshold, the user can send ‘Plus’ from his phone which will increase the fan speed on the leopard gecko. This is indicated by incrementing a counter on the LCD.
* Thus the BLE and the Leopard Gecko perform two-way communication.

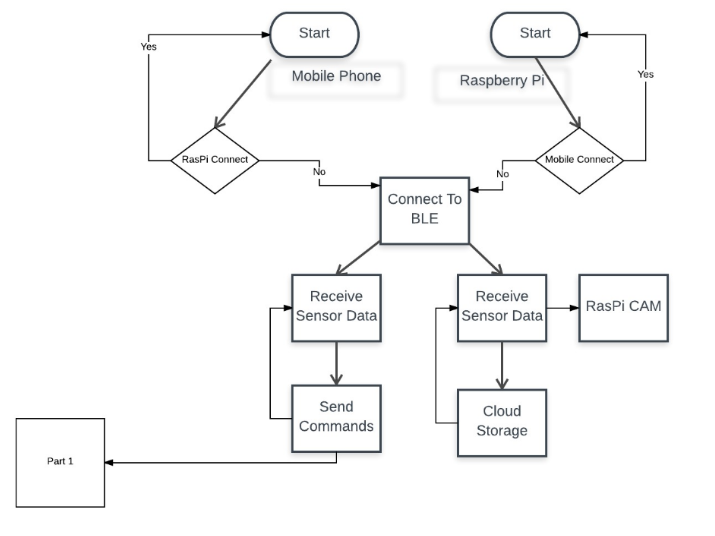
**Raspberry Pi and the Particle Photon:**

* The BLE acts an information provider to the Raspberry Pi which scans for the BLE device every 2 minutes and connects to an advertising Bluetooth LE.
* The BLE then transfers information to the Raspberry Pi.
* The Raspberry Pi has a Bluetooth module connected to it that is used to read the information passed in the form of handles.
* The right handle, containing the information is read and stored in a .txt file.
* This information is then transmitted to a particle photon via UART.
* The particle photon uses an inbuilt api particle.publish() to log the data on google drive.
* Thus, all the data is stored onto a drive for further analysis.
* A camera connected to Raspberry Pi provides the facility of Live video streaming. This helps in observing the condition of the plant growth from remote location.
* Wireless access to Raspberry Pi made available using wireless module (EDIMAX)

**Software Flow Diagram:**



**Figure 4: Software Flow Diagram**

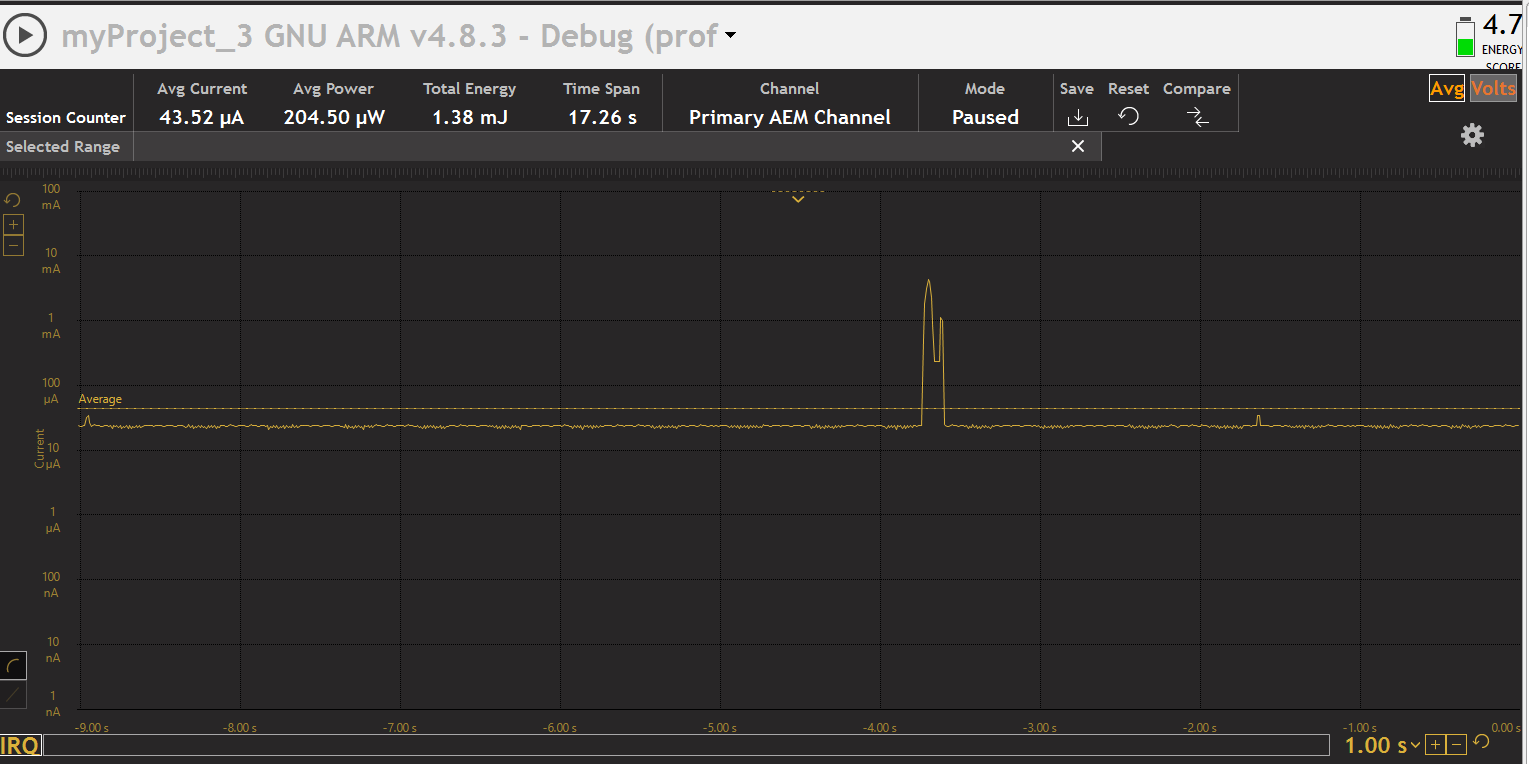


**Figure 6: Project Flow Diagram**

**Test Cases Implemented:**

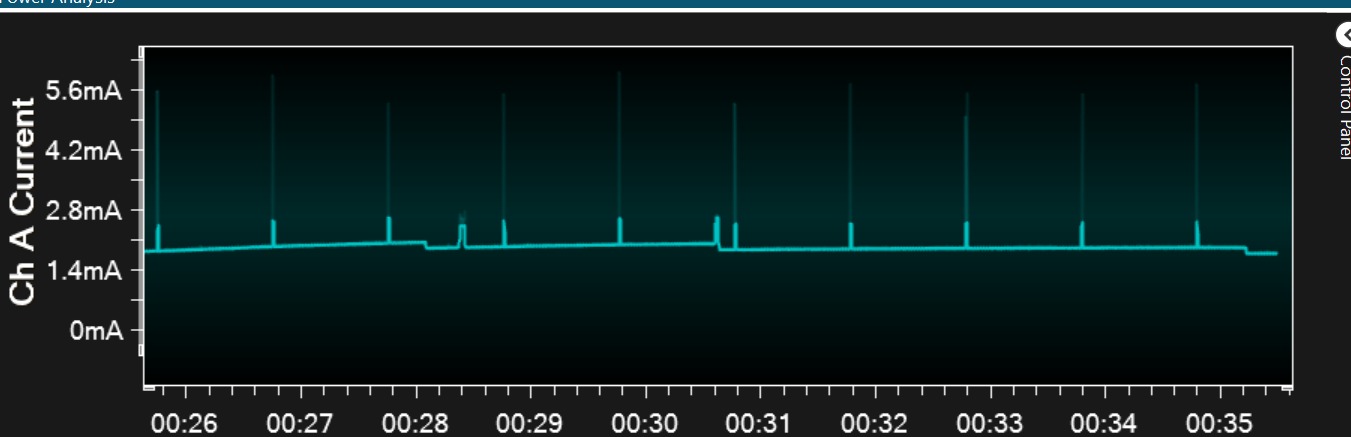
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Function** | **Feature** | **Plan Test** | **Definition of Pass or fail** | **Status** |
| UART Circular Buffer using DMA | Shared Resource | Force 3 back to back writes to the UART | Did all 3 writes complete without data corruption | Pass |
| Cap Sensor | Cap sensor touch controls power to the LG | Does a cap touch show two spikes in energy profiler | Check if all peripherals are disabled if LG powered off by touch | Pass |
| Cap Sensor Power On | Cap sensor touch controls power to the LG | Swipe from left to right to power up | LED goes ON and the Energy profiler works as expected | Pass |
| Cap Sensor Power OFF | Cap sensor touch controls power to the LG | Swipe from right to left to power down | Using LED OFF and also the Energy profiler shows a dip in current consumption | Pass |
| Moisture Sensor | Check values | Check if distinct values generated for moist and dry soil | Is there consistency in the generated readings with a large difference between them | Pass |
| Moisture reading mapping to percentage | Convert readings on a percentage scale | Check if the increase in reading is proportional to % increase | Is there consistency in increase or decrease of percentage | Pass |
| Fluid Level Sensor | Check water Levels | Check if distinct values generated on changing the water levels | Check on consistency by repeating the experiment | Pass |
| Fluid level mapping to inches | Calculate the water level based on digital value generated | Convert digital value to resistance and resistance to inches | Check if inch values are equal to the actual water level | Pass |
| Ambient Light Sensor reading | Check the accuracy of the digital value generated | Cover sensor with a finger to ensure darkness. Then light it up close with a flashlight | Is the value on complete darkness near 0 and value near extreme brightness as high as 0xFFF | Pass |
| Temperature Sensor reading | Check if the temperature readings are accurate | Check if temperature changes marginally | Temperate increases slightly on blowing hot air and reduces on keeping the board in the fridge | Pass |
| RTC | Scan for values every 300 seconds | Check if a spike is generated for 10Sec every 300 seconds | Is the firmware enabling sensors every 300 seconds only | Pass |
| LETIMER0 | PWM on pin PD6 | Change the LED brightness | Checked the LED brightness by changing threshold values coming from Ambient Light sensor | Pass |
| BLE two profiles | Check if two profiles run simultaneously on BLE | Run the chat profile and the temperature profile simultaneously | If the two profiles can successfully display the right data | Pass |
| BLE UART function | Check if the BLE is able to both transmit and receive data | Send data from Leopard Gecko and check on BLE and vice versa | On successful transmission and reception, the respective buffers are updated | Pass |
| BLE Chat (BLE to Phone) | Check if data is transmitted from BLE to phone | Use the Atmel Studio serial terminal to send data | On writing ‘Hello’, a hello is seen on the chat profile | Pass |
| BLE Chat (Phone to BLE) | Check if data is transmitted from Phone to BLE | Use the Atmel Studio serial terminal to receive data | On writing ‘Hello’, a hello is seen on the terminal | Pass |
| BLE Temperature profile | Check the temperature values | Check if the correct temperatures are displayed | Display the temperature on the LG LCD and check if the LCD values and phone values match | Pass |
| Simultaneous communication with BLE | Connect two masters to one BLE | Connect the phone and raspberry pi to one BLE | Check if a connection message is displayed in both the devices | Pass |
| Simultaneous data transfer from BLE | BLE transfers data to both the Pi and the phone | Check if both masters receive the data | If accurate data is displayed on the phone and on the pi | Pass |
| Raspberry Pi Communication | Receive sensor data | Check if data is being sent to the Raspberry PI from SAMB11 | If the sensor data is passed successfully and updated each time the value changes | Pass |
| Wireless Connectivity to Raspberry Pi | Provides wireless communication | Ping google.com by removing the LAN cable attached | Successfully pings data from various servers. Also able to run the browser on Raspberry Pi | Pass |
| Raspberry Pi Camera Interface | Stream video online on remote server | Viewing video with a lag of 2 seconds | Successfully stream video for 10 continuous minutes | Pass |
| Raspberry Pi UART function | Check if the raspberry Pi UART is up and working | Perform a UART loopback and check if data transmitted = data received | On transmitting Hello, a hello is printed back on the terminal. | Pass |
| Integration of Live video capturing and UART Transmission | Check if the integration is working | See the info on cloud and live video streaming in parallel | Successfully observed data and live video | Pass |
| Connect BLE to phone and Raspberry Pi | Check if connection and disconnection works | See if everything goes well | Successful connection and disconnection observed | Pass |
| STOP operation on BLE | After BLE connected to phone disconnect by passing STOP | BLE should disconnect | BLE disconnects successfully and gets connected back again | Pass |
| Receive info after re connection | Check for data validity when BLE is disconnected to phone | BLE should send data in valid format | BLE sends the data for every 5 seconds if connection is maintained | Pass |
| Periodic connection with Raspberry Pi | Raspberry Pi being client should automatically connect and re connect | Check if Raspberry Pi connects and disconnects every 2 minuts | Successful connection and reconnection is done multiple times | Pass |
| Particle Photon UART Function | Check if the Particle Photon UART | Check if the Photon UART receives value from the Raspberry Pi successfully | If the data is received accurately and printed on the terminal | Pass |
| Data logging over the Cloud | Upload all the sensor data using particle photon via the Raspberry Pi | Check if the same data is received on the phone and on google drive | data is stored continuously | Pass |
| Read the value from characteristics | Check if sensor data is read | If the exact sensor numbers are read | Display characteristic information on the command line | Fail |

**Energy Score Analysis:**

****

**Figure 7: Leopard Gecko Energy Profiler**

This energy score was achieved when all the LEDs were off and the capacitive sensor frequency was set to 10.

****

**Figure 8: ATSAMB11 Energy Profiler**

The parameters for connection latency and connection interval were changed to suit the application. The transmission power was decreased so that the BLE has a power range of -70 db in vicinity.

**Lessons Learnt:**

**Issues Faced:**

It is a known fact that when working on a project, maximum time is spent on debugging. Throughout our project implementation, we faced several issues. Here is a list of some of them and how we manage to overcome them:

* **The Cap Sensor swipe speed:**

The capacitive sensor initially recorded only a gradual left to right and right to left swipe for switching the system on and off. Thus, if a user swiped the cap sense faster, the system wouldn’t record it as on or off.

**Solution:** We Increased the scanning frequency from 10 to 50. This affected the energy score but the functionality was achieved.

* **Use of LETIMER for SysTick – PWM conflict:**

Initially, we used the LETimer as our system clock. We also wanted to implement PWM in our project. The onboard LED is connected to Timer3 for PWM. But switching the internal Timer3 on for the entire lifetime of the system greatly affected the energy score. The other option was PWM using LETIMER which was already in use.

**Solution:** We used the RTC clock as SysTick and implemented PWM using an external LED and a GPIO using LETIMER.

* **PWM – I2C0 Pins conflict:**

We used PD6 and PD7 initially as PWM pins. Later we required the I2C0 for the ambient light sensor which used the two port D pins as SCL and SDA lines.

**Solution:** We replaced the PWM pins to be from PortB (10 and 11) to avoid conflict.

* **Connecting to a phone and Raspberry Pi simultaneously:**

The biggest challenge in this project was to make the BLE communicate to both the phone and the Raspberry Pi together. We had to do a lot of research to figure out all the possible options that we had to implement this function.

**Solution:** We first decided to use an inbuilt multirole profile that made the BLE to operate as a slave and as a master alternately. The disadvantage there was that for the BLE to operate as master, it can only can for other BLE devices. We had a raspberry Pi with a Bluetooth classic module and hence we eliminated this option. Next, we tried to perform indirect advertising for the phone and then direct advertising to the raspberry pi alternately. This seemed to be a good option initially. The only issue was our raspberry pi detected the device MAC ID but couldn’t recognize the profile in direct advertising. Thus, we performed indirect advertising for both devices where the Raspberry scanned for a BLE every 2 minutes and disconnected in 10 seconds. The user could connect anytime and to disconnect, the user had to send a ‘stop’ command on the chat profile and the BLE the started advertising again.

* **Reading handle information at the Raspberry Pi:**

We were able to extract all the handles using the HCITOOL. But the issue we faced was that we couldn’t read the handle that contained all the information (handle #0008) because each time we tried doing that, the Bluetooth connection would just stop responding. Because we were new to the Raspberry Pi platform, we couldn’t find a way to extract out the data and we could only extract the connection name ‘AT-CSC’ from the handle.

* **Raspberry Pi to Photon Connection:**

We explored several options for exchanging information like using USB or SPI communication but because we weren’t experienced enough with Raspberry PI, we decided to stick with UART for communication. However, the Raspberry Pi UART is tied to the console for executing print functions.

**Solution:** We had to follow certain steps to disable the console to enable the use of UART for communication.

* **Chat profile – Temp profile conflict**

We implemented both the chat and the temperature profile together. But every time we opened the temperature profile on the phone, it would freeze the chat Rx line which would work perfectly otherwise. We found the error to exist because of a parameter in the htp\_init function that wrote temperature as characters. This would then stop any other character from sending information.

**Solution:** We replaced the parameter and tried all other possible options including a NULL

* **Common Ground Problem**

We were unable to communicate the Leopard Gecko and the ATSAMB11 once. After checking the UART connections, we started checking our code for errors. It was later that we realized that both the boards require a common ground for communication.

**Solution:** Now we either connect both boards to one PC or use a common ground using breadboard

**Learnings:**

The whole purpose of taking up a project as complex as this was to learn a lot of things and understand BLE concepts in and out. We are glad we could implement most of the features proposed in our project proposal. We surely learnt a lot of things from this experience.

* We learnt the process of integrating several sensors onto a leopard gecko and making them to work simultaneously. We realized that the sensors sometimes work well when they’re tested alone, but they can give you hard times on integration.
* We learnt the implementation and realized the importance of having a circular buffer in the project and also the use of a DMA to control data exchange to save CPU power.
* The most important thing that we learnt in the project was **‘Bluetooth Low Energy’**. Thanks to our application, we now know the ATSAMB11 and BLE concepts in and out. We’ve spent most (about 60%) of our project time on the BLE and made it to perform communication using direct advertising, indirect advertising, BLE as slave, BLE as master, BLE with multiple profiles and BLE with simultaneous connections.
* Both of us had never worked on a Raspberry Pi platform before. We were eager to learn about it and hence we decided to use it as our sensor hub. Though we faced several difficulties which someone with previous experience wouldn’t have faced, we wanted to learn as much as we could and we did that satisfactorily.
* We also learnt the importance of coding from a **‘Low Power’** point of view. Earlier, we used to write firmware only from the functionality point of view. This project, and this course as a whole, has helped us think about firmware from the power consumption point of view. We now code using the principle **‘keep devices ON only for the time that they’re required and shut them OFF again’.**
* This project also gave us an opportunity to work with the Particle Photon board which according to us is a pretty smart board for IoT applications and we intend to explore it even more in the future.
* Lastly, we improved our **Debugging Skills.** Because of the many issues that we faced throughout our project, we surely improved our debugging skills and our approach to find an error in the code. Overall, we’ve tremendously improved our coding skills and that was the whole purpose of our project.

**Future Expansion:**

We’ve left a lot of areas open for expansion. We are actually going to implement the project as a complete product under Professor Corelle from CSEL LAB and Ryan Woltz from EDN. These are the things that we intend to produce under them using the product that we’ve built for this course.

* We intend to implement several other sensors like the CO2 sensor, the humidity sensor and the pressure sensor to another Leopard Gecko board as we used up most of the GPIO provided on one board.
* At the BLE, we intend to create a separate profile for each sensor value by creating a custom BLE profile. We intend to use the chat profile to only send back commands to the sensor node.
* The Raspberry Pi being the Sensor Hub, will communicate with more nodes at once and log all the data onto the cloud.
* The camera interfaced to the Raspberry Pi will perform more complex image analysis and raspberry Pi will also have the provision to send notifications to the user if images change drastically via the BLE
* Thus, we intend to add more and more automation for this to be a successful product idea.

**Conclusion:**

Thus, this project proved to be extremely useful as it helped us learn several skills and techniques required in higher architectural programming. Reading several user manuals and datasheets allowed us to enhance our analysis skills. We thoroughly achieved our goal of learning immensely from this project. We would like to thank several people in guiding us or helping us with their inputs.

Firstly, we would like to thank our TAs Shiva and Omkar who helped us figure out Bluetooth connection with Raspberry Pi from their previous experiences. Omkar was kind to give his raspberry Pi board to us for the project. We would also like to thank Rishabh Berlia from the previous batch who implemented an idea similar to ours. Lastly, we would like to thank Mounika Reddy from our batch who helped us in debugging when we were stuck with direct advertising implementation.

We would like to thank **Professor Keith Graham** for all is inputs. We would’ve spent many more hours in debugging our issues if it wasn’t for his help. We approached him several times with our issues and he was always kind to help.

In conclusion, we tried to build a project from a product point of view, and this project idea, when implemented on a larger scale can be a boon for plant life.

**References:**

* https://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32LG-RM.pdf
* http://www.atmel.com/images/atmel-42426-smartconnect-samb11-soc\_datasheet.pdf
* https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf
* http://www.raspberry-projects.com/pi/programming-in-c/uart-serial-port/using-the-uart
* https://docs.particle.io/reference/firmware/photon/
* http://asf.atmel.com/docs/latest/applications.html
* https://learn.adafruit.com/install-bluez-on-the-raspberry-pi/overview
* <https://cdn-shop.adafruit.com/datasheets/eTape+Datasheet+12110215TC-12_040213.pdf>

|  |  |
| --- | --- |
| Harshil Sheth  Harshil.Sheth@colorado.edu  Department- ECEE  University of Colorado, Boulder. | Rahul Yamasani  Rahul.Yamasani@colorado.edu  Department- ECEE  University of Colorado, Boulder. |

**Project Staffing:**

**Appendix:**

**Raspberry PI Setup Steps:**

**Step 1: Boot up the SD card with Raspian Os**

Install rasbian os into SD card using Win32DiskImager

Follow This link:

<https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

**Step 2: Finding ip address using nmap**

Connect pi to computer and Ethernet

Find the ip address of pi using nmap –sn [ip address/MASK]

**Step 3: SSH pi**

Open Putty and ssh using the ip Address found (All set for ssh till here)

**Step 4: Connecting to WIFI using edimax**

<http://raspberrypihq.com/how-to-add-wifi-to-the-raspberry-pi/> (very very good tutorial)

Now find the ip address for wlan0 using “**ifconfig wlan0”**

**Ssh** using the wlan0 to ip address using putty. You are all set to connect wireless to your Pi

**Step 5: vnc**

Open RPi terminal

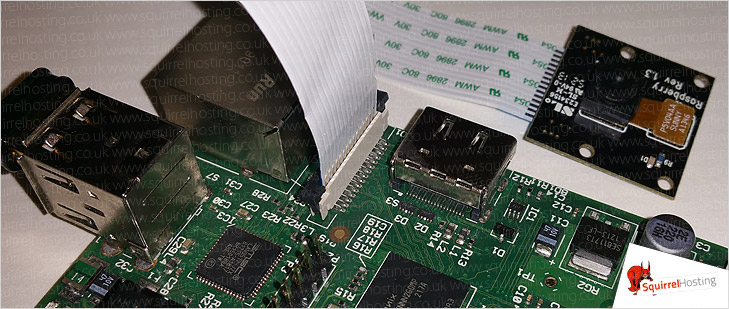
Install and run **tightvncserver :1** (space before semicolon)

Open tightvncserver and connect using the [ip address:1]

NOTE: you can also use REALVNC here

**Step 6: Connecting Rasberry Pi cam**

Connect the cam with blue label towards Ethernet port



RPi cam tutorials:

<https://www.raspberrypi.org/learning/getting-started-with-picamera/> (python code to interface with cam)

<http://elinux.org/RPi-Cam-Web-Interface> (make raspberry pi cam as server camera)