FPGA Enhanced Wireless Sensor Node for IoT Applications

Team
Ishak Nur
Logan Robson
Jesse Esquivel
Mohammed Marjan
Hung Tran
Irfan Malik

Faculty Advisor
Dr. Jens-Peter Kaps

Executive Summary

Unfortunately, the Internet of Things(IoT) is not perfect. Up to date there has been numerous attacks on IoT devices, leaving users vulnerable to data & identify theft, eavesdropping, blackmail, etc. Securing the IoT is a primary concern and one that must be resolved to fulfill the IoT paradigm. Securing the IoT is a broad topic and involves expertise in many fields. In this document, we focus on securing communication between an IoT device and a gateway device while aiming to increase power efficiency of a Wireless Sensor Node (WSN), which plays the role of an IoT device in our application. We also devise simple, but effective methods to achieve an easy user interface along with other requirements to ensure a successful design.

Table of Contents

1. Approach

1.1 Motivation

Internet of Things (IoT) is the global phenomenon of connecting homes, cars, wearables, and much more to the internet. Today, IoT is evolving at an alarming pace in terms of the functionality of applications, usability, and number of devices being connected to the internet. While this simplifies the lives of many using IoT applications, it also brings about unavoidable issues that must be tackled for the IoT to take off and be fully accepted. Security is arguably the biggest challenge facing the advancement of IoT, flaws in security can have major consequences, jeopardizing the user's privacy, identity, and data , possibly leading to financial risk. Power consumption of IoT devices is another major concern. IoT devices such as wireless sensor nodes are devices that sense for specific targets in their physical environment, digitally process what they have sensed, and finally transmit digital information that can be easily understood to an end user. WSNs are usually battery powered and deployed in all kinds of environments and places, and so a need for the devices to have low power consumption arises to increase battery life

1.2 Solution

In our project, we offer a solution to many security vulnerabilities that are prevalent with IoT devices, as well developing a Wireless Sensor Node that focuses on lower power operation and design. As there are many applications a wireless sensor node can take, our specific example for development and proof of concept is a wireless security camera. The basic function is that when motion is detected within a certain area, an image is captured and sent to the user via an android app. The node contains a microcontroller for reading and sending information and commands to the sensors and the user, a camera, PIR motion sensor and an FPGA for security enhancement. The Gateway, consists of a BeagleBone Black, which communicates information to and from the user such as images and data requests from the node.

When designing our wireless sensor node, we concentrated our selection of components on ones that would achieve the greatest balance of power consumption and performance. However only selecting efficient components is not enough, we also needed to integrate them in a way that takes advantage of their benefits. Such as with our microcontroller selection, when the microcontroller is not in an active, processor intensive mode, we power it down to the lowest power operation possible while still providing the functionality we need. Additionally, the FPGA is turned off when not being used, also reducing the amount of power consumed. And finally, our transmission and receive devices were selected such that they offer the greatest balance of data rate, range and security while consuming the lowest power and offering a various sleep modes.

As we stated previously, security is one of the main vulnerabilities for IoT devices. We alleviate this issue by securing all the devices and channels in our wireless security camera. There are two main issues with security, one is making it easy for the user to setup said security such as device authentication and the other is securing devices with complex keys that cannot be easily deciphered, while also having a means to update and change keys frequently. We achieve this by first allowing the user to easily setup up a node in the Wireless Sensor Network, via QR code in our android app. The data from the QR code is uploaded to a Dropbox server via a secure login, and

then retrieved onto the gateway for use.

This QR code contains the node's public key and the node's address, this information is used to start our encryption algorithm as well as adding the node to the network. On the gateway device, a session key is generated using the Elliptic Curve El-Gamal encryption scheme. The scheme also generates points needed to decrypt the session key. These points are then sent over to the Node, which contains a microcontroller and FPGA. The FPGA follows the same El-Gamal scheme and uses these points to decrypt the session key. Now any data sent between the Node and the Gateway is encrypted with the communication modules built in 128-bit AES but using the decrypted session key as the key for AES. An extension of this is that if more Nodes for different purposes besides a wireless security camera are added to the network, they will have different session keys. Therefore, in the case of a security breach if one component is compromised, other components in the network are not.

1.3 Alternative Designs

1.3.1 XBee vs CC2530

Initially we decided on our communication device to be the CC2530, which also uses the ZigBee 802.15.4 protocol. However, after receiving the device we found it difficult to program and implement all the features we would like the modules to have. The setup was very involved, which included flashing the modules with Z-stack software among many other things. Therefore, we decided to go with the XBee modules, which still uses the ZigBee 802.15.4 protocol, but takes care of the underlying software for you. Thus, allowing us to focus on the other aspects of designs, while also implementing a design that is scalable for future development.

1.3.2 Camera Module

One of the alternative designs we considered, specific to the demonstration of our system, was the choice of camera module. The CMOS sensor we are using comes in two main varieties, the first of which is a breakout board with a limited number of resistors and capacitors, and is interfaced via I2C to change the sensors settings, and parallel 8-bit data output of image data, with VSYNC and HREF signals to delineate image frames and lines, respectively. The CMOS sensor produces a new 8-bits of data at a minimum frequency of 6Mhz, which is too fast for many low power microcontrollers. The second variety seeks to alleviate this problem by including an on-board FPGA and FIFO memory. The FPGA is used to read in image/frame data from the image sensor into the FIFO memory, and provide a SPI interface to the FIFO, so that external devices, too slow to read in the image/frame data unassisted, can receive the image/frame data via a SPI interface. The first variety (without an FPGA or FIFO) costs around $10, while the second variety costs around $25.

In our design, we worked with both varieties, with the intention of achieving the lowest power consumption at the lowest price. The microcontroller in our design is too slow to use the first variety (without an FPGA or FIFO) to read in image data, so we implemented a similar design to the second variety, using the FPGA on our board, as well as a Cellular RAM, however due to time constraints on our project this implementation was not fully realized, and we decided to use the second, more expensive, variety of the camera module. Ideally, our design would use our on-board FPGA along with a SDRAM and the cheaper camera module, to achieve identical results to that of

the more expensive camera module.

1.4 Team Member Contributions

1.4.1 Hung Tran

User Interface

- Develop an Android app as user interface
- Implement the QR droid library in QR code scanner function.
- communicate between the app and Dropbox. Also, firebase and the app.
- implement the Dropbox API in android app.
- Send user request from the app to node.
- Send session key change from the app to BeagleBone Black
- App gets an alert notification from firebase when image is uploaded and allows user to view image

BeagleBone Black

- Communicate between Dropbox and BeagleBone Black include download, upload, and delete files.
- Detect new images in BeagleBone Black.

Cloud

- Connect Dropbox and Google firebase backend.
- Detect new files on the cloud.

1.4.2 Logan Robson

Arducam OV2640 Camera Interfacing with MSP432

- Initialize OV2640 CMOS sensor via I2C using arrays of address, value pairs to be written to the OV2640 registers.
- Use SPI to communicate with the Arducam to send command to capture image, poll the Arducam via SPI for the capture done bit, burst read image data from Arducam to MSP432

XBee X2C Interfacing with MSP432

- Use UART to do basic transmission and receiving between the XBee modules.
- Use UART to enter "Command Mode" of the XBee module, as well as code to configure the XBee module using several different AT commands.

MSP432 C Code Misc.

- Code to interrupt/ISR for PIR motion sensor interrupt.
- Code to enter a low power state, interrupting every 30 seconds using the RTC on the MSP432, and waking the XBee module to perform polling of the XBee coordinator.

Schematic and PCB

- Created schematic for different components needed for MSP432 and Actel Igloo FPGA, as well as connectors and components for Arducam OV2640, HC-SR501 PIR and XBee S2C.
- Designed PCB using KiCad/PCBNew.

1.4.3 Irfan Malik

XBee Interfacing with MSP430 and BeagleBone Black

- Transmit and receive data using UART with the XBee module
- Enter XBee "command mode" via UART, to setup security, sleep, coordinator, end

device etc.
- Upload images to cloud with timestamp

BeagleBone Black

- Send and receive information between the two XBee modules individually
- Setup gateway to transmit images from node to cloud
- Read user requests from cloud and execute desired actions based on request
- Delete files on cloud and BBB after use
- Run El-Gamal encryption scheme on BeagleBone Black to generate session key for AES on XBee

1.4.4 Ishak Nur

- Debug, update, and setup peripherals on the BeagleBone Black gateway device.
- Researched and learned about Elliptic Curve Cryptography and its various implementations.
- Assisted Jesse in getting ECC core to work on FPGA.
- ECC on the BeagleBone Black
    - Implement Point Addition, Point Subtraction, Point Doubling, and Scalar Multiplication functions (Open source code used from [12]).
    - Implement the Encryption part of an El-Gamal ECC Cryptosystem, which works with FPGA (Got Assistance from [16]).
    - Update session key upon user request
- Final Presentation Slides & Diagrams.

1.4.5 Mohammad Marjan

- Power the circuit that will convert from 5V to precise voltage needed to power different parts for our circuit node.
- Motion sensor circuit using MSP432
- Prototyping motion sensor
- Testing motion sensor

1.4.6 Jesse Esquivel

MSP430 Interfacing with Nexys 4 Board

- Wrote C code for the MSP430 to act as a master in I2C communication during testing phases. Data gets sent and receive using interrupts.
- Modified Open Source I2C Slave Unit[6] in VHDL in order to better fit our needs to communicate between microcontroller and FPGA.
- Created a transmit/Receive Unit in VHDL to facilitate data transfer to and from the ECC core.
- Took care of testing/debugging VHDL design using Xilinx ISE Simulator for test benches during testing phases.

MSP430 interfacing with Actel Igloo nano Board

- Port VHDL code onto Libero SoC.
- Perform final testing/ debugging on Actel Igloo nano FPGA starter kit for design.

Cryptography

- Researching the field of Cryptography, specifically Elliptic Curve Cryptography.

2. Technical Section

2.1 System Models

2.1.1 System Architecture

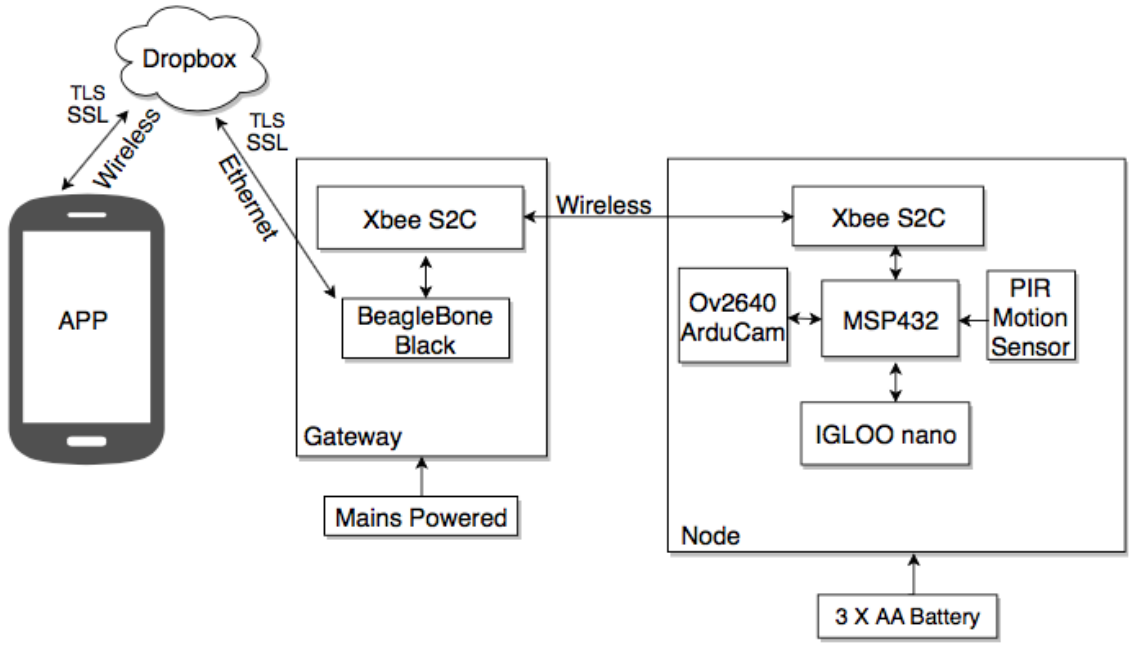

Figure 1 - Overview of System Architecture

Figure 1 shows our overall system that includes our App, the Gateway, and the Node. The App connects to the Gateway via the cloud using both wireless and wired communication. The gateway connects to the Node via wireless communication. The Gateway is connected to Mains power as it is intended to be constantly on, while the node is battery powered.

2.1.2 Functional Decomposition

2.1.2.1 Top Level

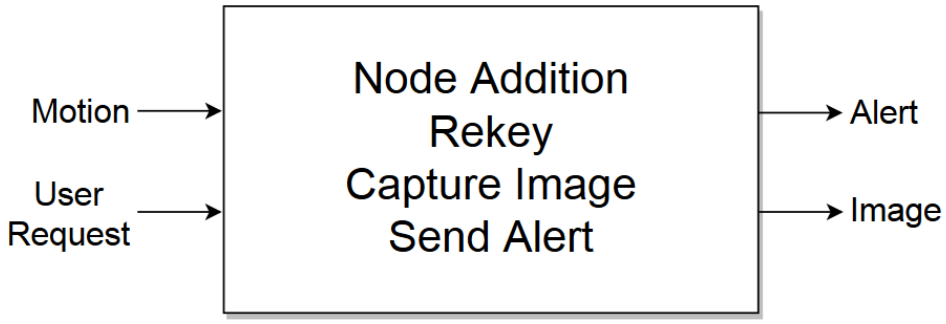

Figure 2 - Top Level Design

Figure 2 shows our top most level. This top most level includes four main functions that our overall system performs. As inputs into the overall system we have motion and user request. Motion sensor comes from the PIR on the sensor node while the user request comes from the buttons on the App. As outputs from the system we have an alert notification and image captured both which get sent back to the App.

2.1.2.2 Functional Breakdown

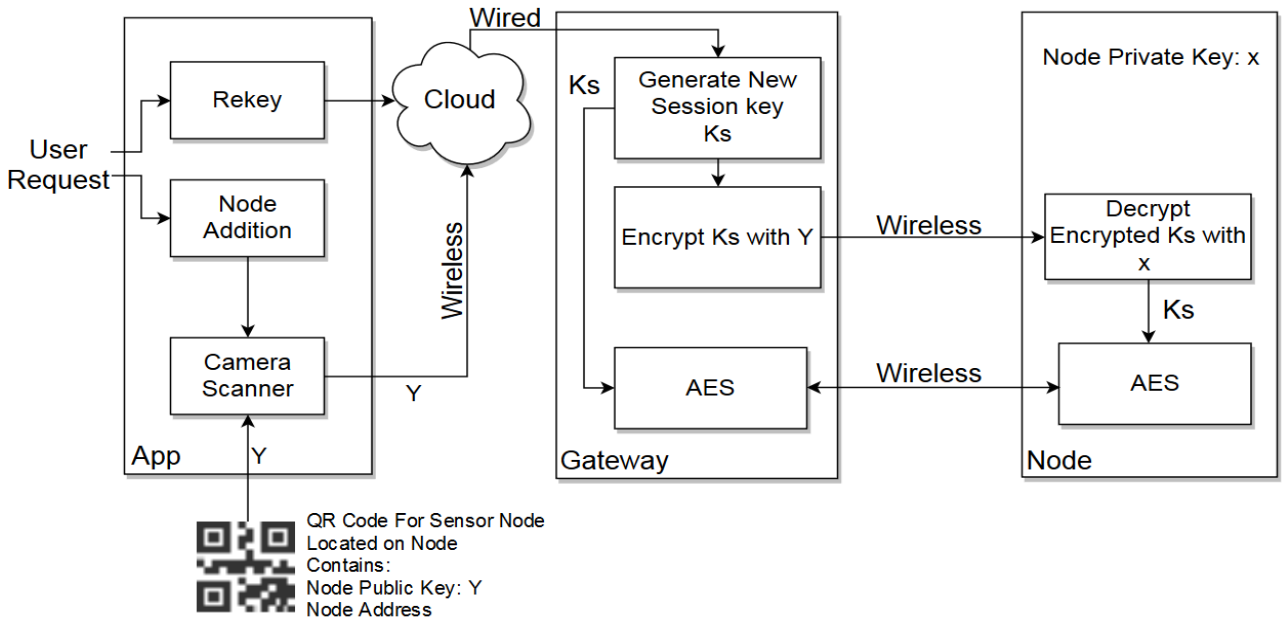

Figure 3 - Node Addition & Rekeying Functions

Figure 3 shows a deeper insight into how our system works, Specifically for our Node addition and Rekeying functions. We see two user request buttons: Rekey and Node addition. In order to have insert a node into the network, the app scans the QR code on the sensor node which contains information such as Node Public Key and Node Address both of which the Gateway needs. With this information, the Gateway can now generate a session key and encrypt it passing it along to the sensor node. The sensor node decrypts the session key as well. Ultimately both the Gateway and the Node pass the session key into their respective Radio modules which contain AES encryption. It creates a secure channel between the Gateway and the Node.

2.2 Description of System Elements

2.2.1 Elliptic Curve Cryptography(ECC)

Elliptic Curve Cryptography is a public key cryptosystem and is a strong alternative to other cryptosystems such as RSA because of the fact that it uses smaller keys and ultimately requires less storage and less memory. ECC is based off elliptic curve arithmetic. An elliptic curve has many points within a finite field. Basic operations performed on these points include point addition, point doubling, and scalar multiplication which is essentially adding a point to itself k(scalar) times. Now, how exactly did we use ECC? We implemented an El-Gamal scheme specifically using the ECC operations mentioned to generate a session key, encrypt it, and finally decrypt it.

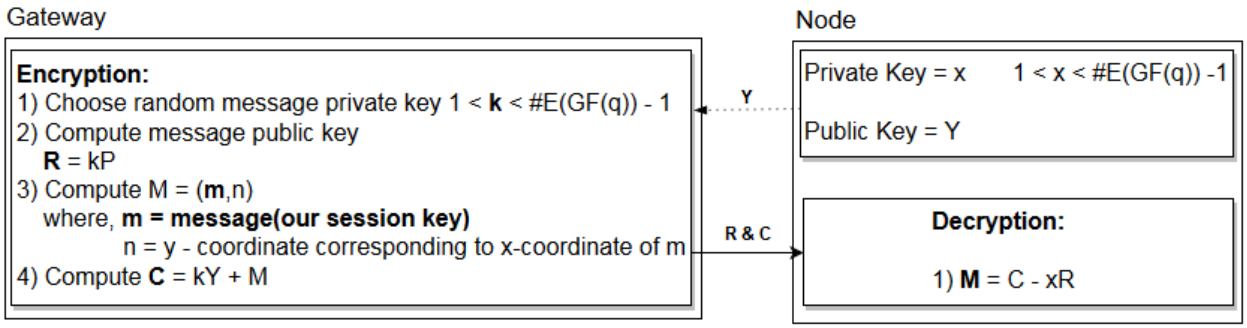

Figure 2: Elliptic Curve El-Gamal[15]

Proof:

$$C - xR = (kY + M) - xR$$

$$= (kY + M) - xkP$$

$$= (kY + M) - k(xP)$$

$$= (kY + M) - kY$$

$$= M$$

m: x-coordinate of M is our session key

As we can see through the diagram and through the proof, The value of "M" is able to be securely distributed between two parties such as in our case with the Gateway and Node.

2.2.2 FPGA

The FPGA was an essential part to this project as it was used to perform heavy duty ECC point arithmetic computation. Examples of point arithmetic have already been mentioned in the previous section. Ahmad Salman, a graduate student and member of the Cryptographic Engineering Research Group(CERG) at George Mason University provided us with an ECC Core, written in VHDL, which performs the correct point arithmetic operations needed.

We specifically chose the ACTEL IGLOO nano AGLN250V2-VQG100 as our design FPGA due to its low power and flash-based characteristics. Because we only used the FPGA when needed, such as when adding a node to the network and rekeying, the majority of the time it is turned off. The flash based characteristic of the FPGA was beneficial as upon every power up of the FPGA our VHDL design was already stored in the FPGA meaning we did not have to manually program it. During the testing phase we used the Actel Igloo nano Starter kit[8] to validate and ensure our VHDL and ensure that our overall design was able to fit within the FPGA size limits and resources.

2.2.3 Microcontroller

The microcontroller was used for the processing in the sensor node, and to facilitate communication between the different peripherals and components of our system. Specifically, the microcontroller remains in a low power state, and wakes on periodically using the RTC or via external interrupt from the PIR sensor. On an RTC interrupt, the microcontroller checks the receive buffer from the wireless communication to process any pending request, if needed. During the setup and rekeying phase, the microcontroller handles communicates from to and from the wireless communication module to the FPGA. The microcontroller receives points generated from the El-Gamal encryption scheme coming from the Gateway, which is then passed to the FPGA via I2C, decrypted to a session key, and passed back to the AES module of the wireless communication module via UART.

2.2.4 Wireless Communication

The wireless communication devices are used to send the image data from the node to the gateway, which then uploads the data to Dropbox, or convey user requests from the gateway to the node. The data is sent from one module to another with a 9600 Baud rate and using UART protocol. We selected XBee devices for communication for many reasons that comply with the requirements of our project. One of which is power consumption, the XBee modules can be placed into a sleep mode during which they are sleeping for a certain amount of time, and are waken up to check for input, then go back to sleep. This sleep state uses very small amounts of current draw thus prolonging the life of the Node. Our current setup, which can be changed depending on application, sets the node transceiver module to sleep for 50 seconds, then wakes up for 10 seconds to see if any requests were sent from the gateway or if the Node itself needs to send something to the gateway, then repeats this process. However, since we want to send data when motion is detected, we also connected a wake-up pin, which is triggered when motion is detected, which wakes up the Node transceiver if sleeping, sends the data to the gateway, then enters sleep mode again.

Additionally, XBee provides AES encryption with 128-bit keys, we generate these keys from our ECC El-Gamal encryption scheme and set the AES session keys for the modules to these values. Once the communication is encrypted no other device can receive or send data to the modules. To allow for more dynamic key changing we added an option where the user can generate keys for the AES encryption periodically, this is done remotely by entering the command mode of the XBee modules, and giving the gateway and node module the new generated session key, however all the user needs to do is click the "new key" button in the android app.

2.2.5 Sensors

The two sensors in our design are a PIR sensor, as well as a CMOS image sensor. While the sensors themselves are a part of our design, the microcontroller does not interface directly with the actual sensor, instead with peripherals on the breakout boards of the particular sensor. The PIR breakout board features a BISS0001 PIR controller which interfaces to the microcontroller via a GPIO pin. The breakout board for the CMOS image sensor features an FPGA and an FIFO for storing image and frame data. The FPGA provides a SPI interface to the FIFO, which the microcontroller uses to read in image data. Additionally, the CMOS sensor features an I2C interface, which is used to set the setting registers of the image sensor.

2.2.6 User Interface

We created an android smart phone app to function as our user interface. The app is developed with two layers: a main layer and a sub-layer. The main layer four buttons to perform four functions such as set up a new node, access cloud storage, image request and change session key. The second sub-layer is used for potential functions in future use. The setup node function opens a QR code app to scan the node's QR code and uploads the data to the Dropbox cloud. The access cloud storage function allows the user access to the Dropbox cloud to see all the data. The image request function generates a file to send to the node through Dropbox and the gateway device. The last function, change session key, is used to signal the gateway and node to generate a new session key for the AES encryption. The user is alerted of new data in two ways. The first way is by an in-app notification which is executed by the cloud when a new image appears on the cloud. The second way is by text message alert, the gateway sends a text message to a phone also when a new image is uploaded to the cloud.

2.2.7 Gateway

The BeagleBone Black is the central hub of our IoT application, it connects the node to our smart phone App. The purpose of the Gateway device is to listen to both the Node and cloud awaiting commands and managing data traffic. From the Node, the Gateway intakes an alert and an image once motion is detected, which is then pushed to the cloud. From the cloud side, the Gateway awaits commands from the App to either request an image to be taken, add a Node to the network, or change session key between Node and Gateway as described earlier.

Another key role of the Gateway is how it plays part of an Elliptic Curve El-Gamal scheme. The Gateway generates a session key that will be known to both the Node and Gateway. In the Gateway, we pass the session key into the wireless communication module with AES capabilities. Also, this scheme is setup so that it can easily be run again to generate new session keys. The Node

decrypts the session key as mentioned in the microcontroller section. This allows for data between the Gateway and Node to be encrypted with advance keys that can be updated making this channel even more secure.

n the Gateway, we used python in order to deal with the encryption side of the El-Gamal scheme. With known parameters such as Node public key, field, and curve parameters we could perform the operations shown in Figure 4. In regards to the session keys, we used a random number generator algorithm in order to generate them. Fortunately, there were many open source implementations and the functions that perform these operations which were obtained from [12]. The written code is heavily interactive with an App that allows a user to update session keys.

2.3 Physical Design

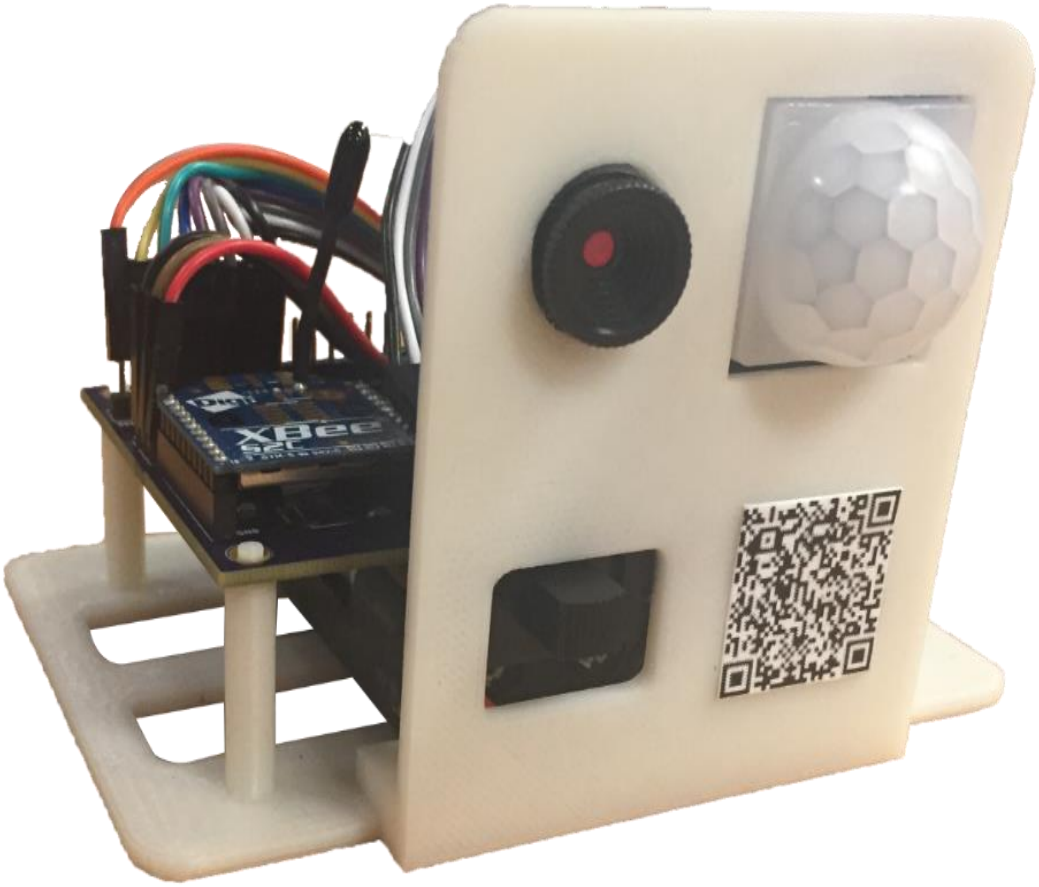

Figure 5 - Final Design of our Physical System

2.3.1 Schematics

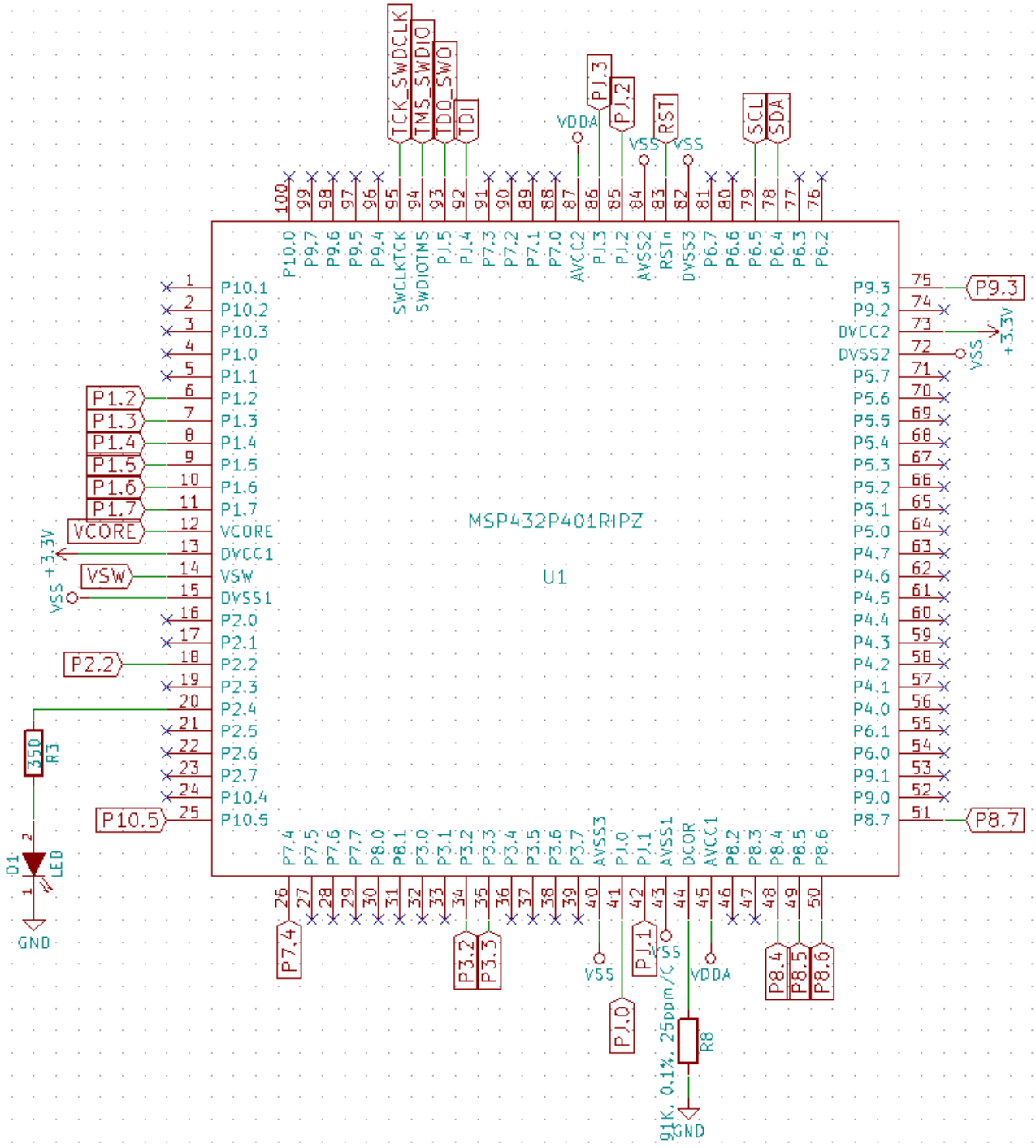

Figure 3 - Schematic of MSP432

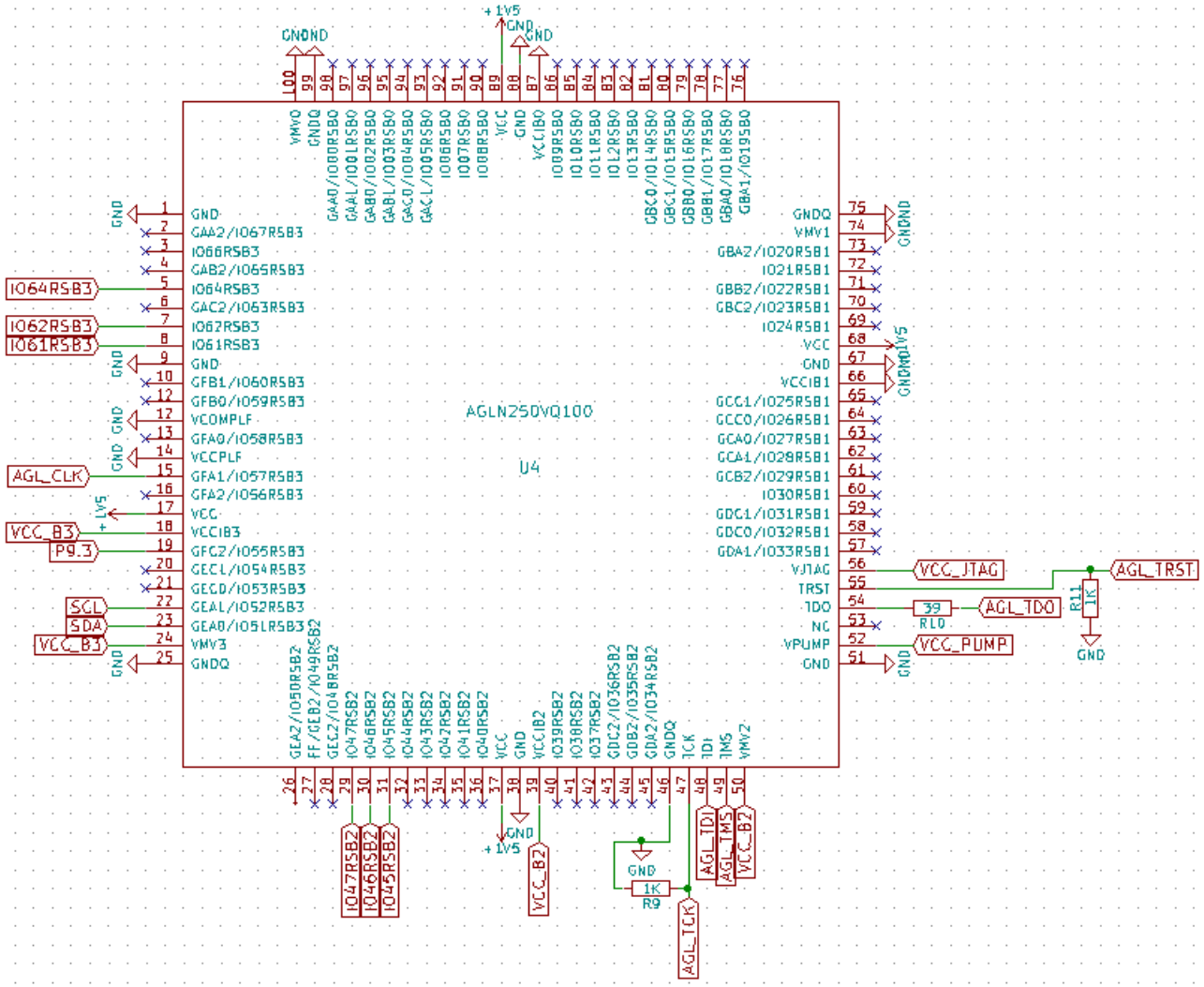

Figure 4 - Schematic of Actel Igloo nano

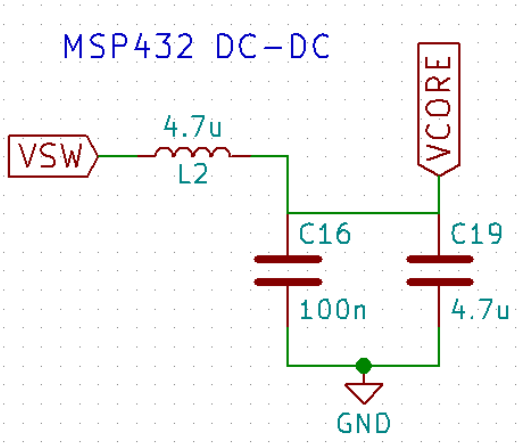

Figure 8 - MSP432 DC-DC

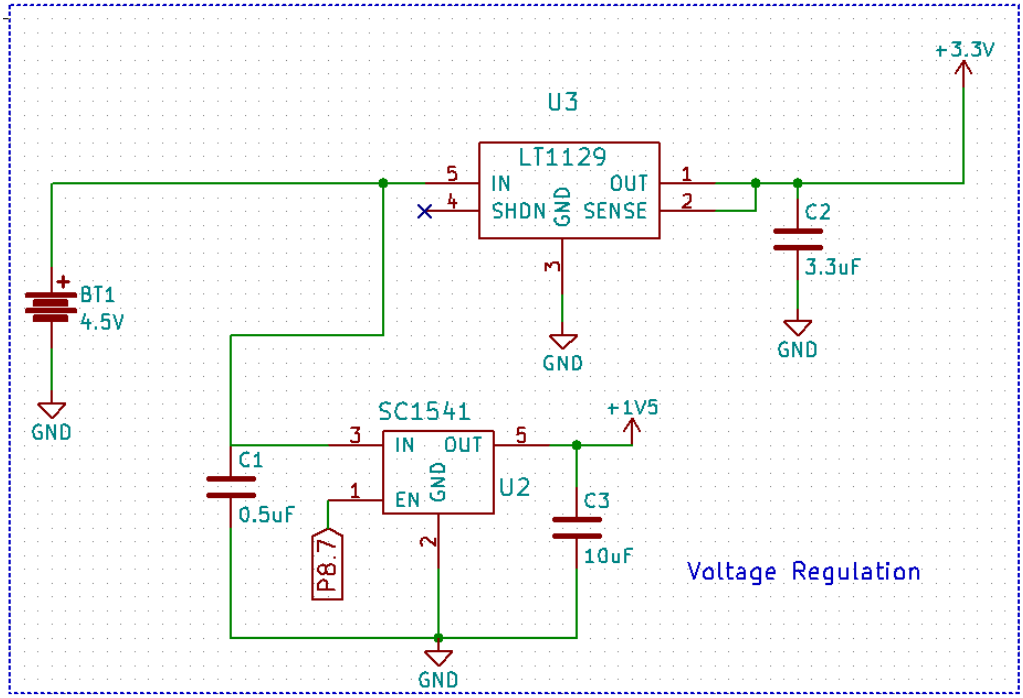

Figure 9 - Voltage Regulation Circuit

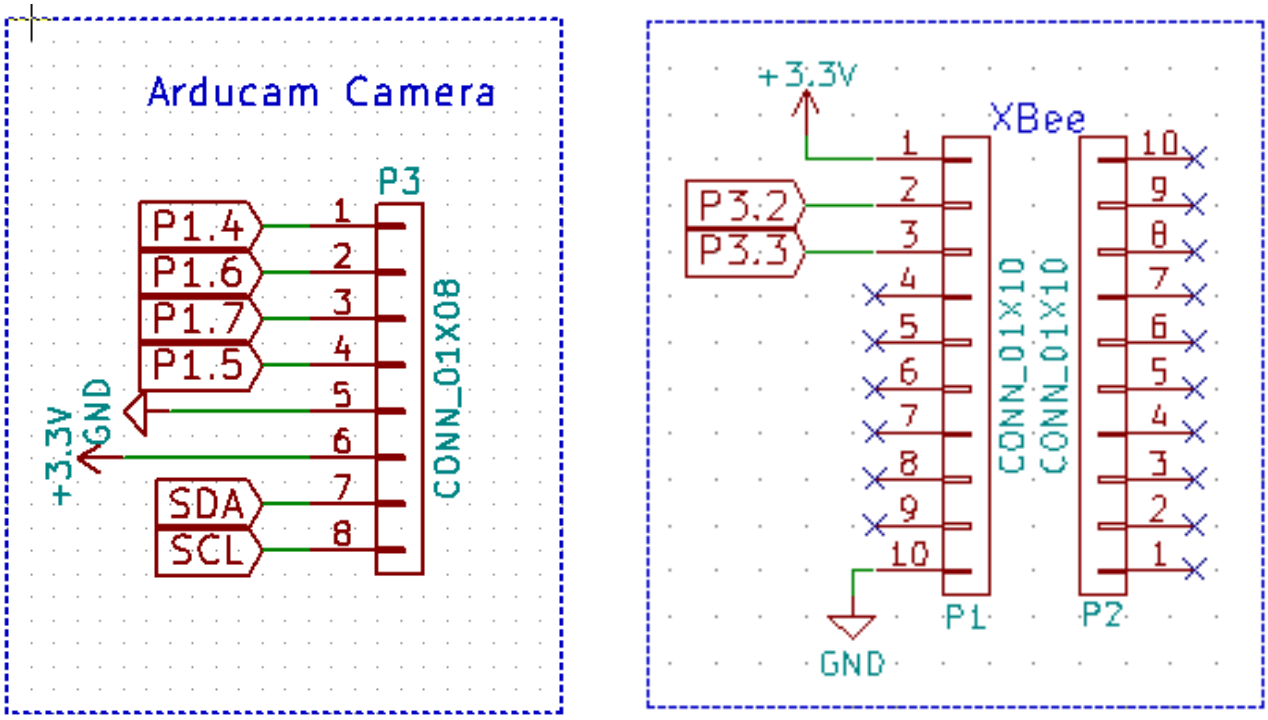

Figure 10 - Pin Headers for Arducam Module and XBee Module

2.3.2 PCB

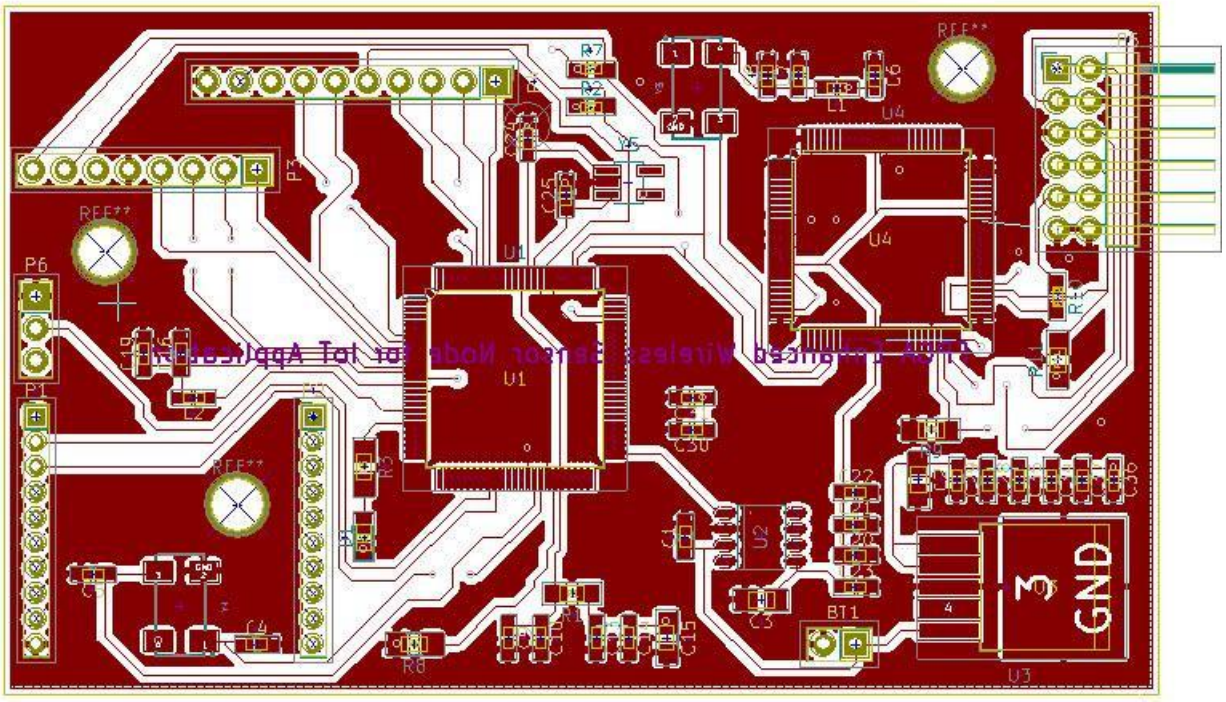

Figure 11 - Front PCB Layout

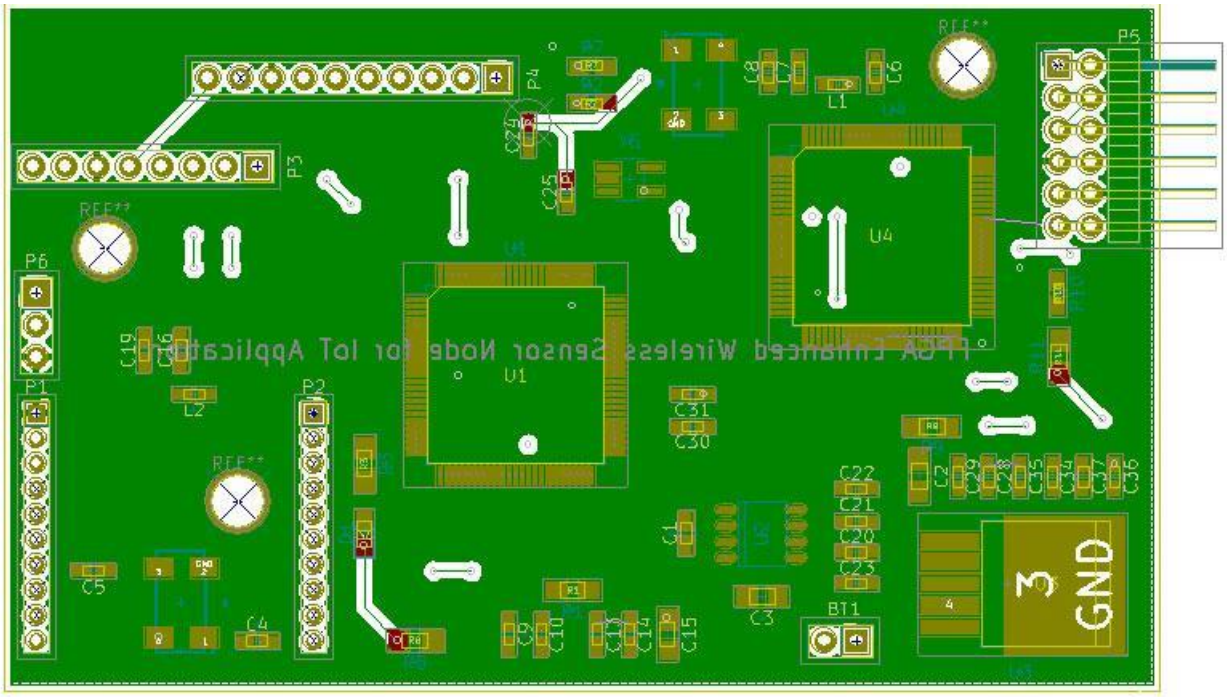

Figure 12 - Back PCB Layout

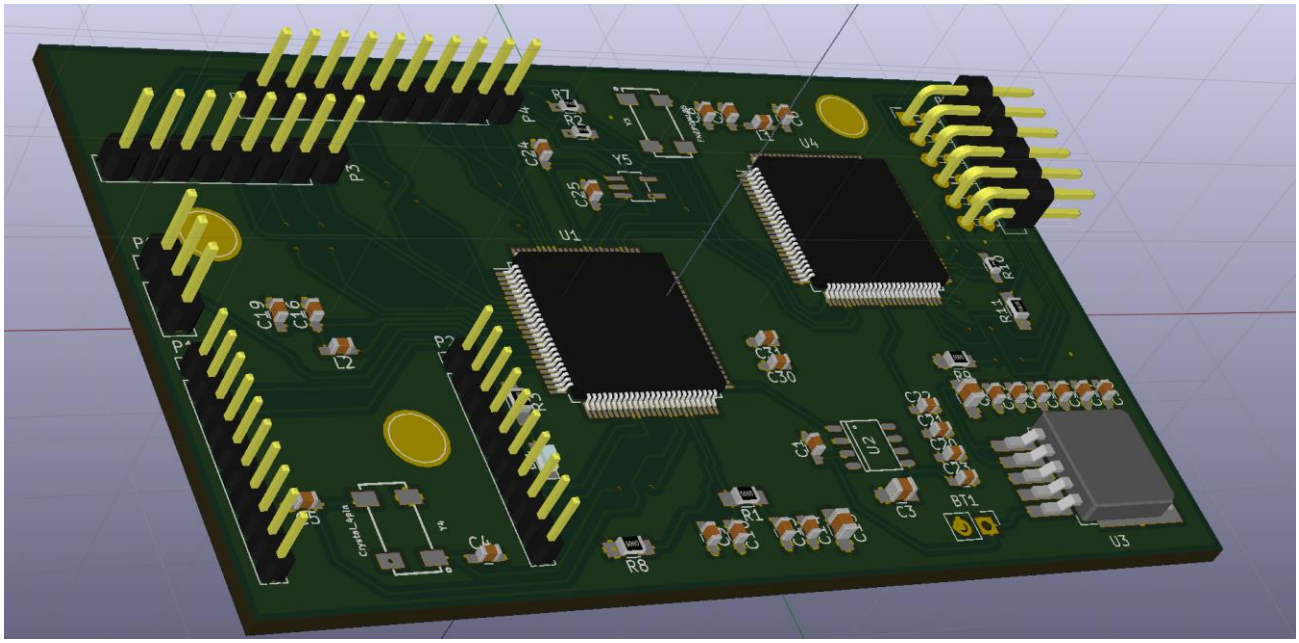

Figure 13 - 3D Generated Model of PCB

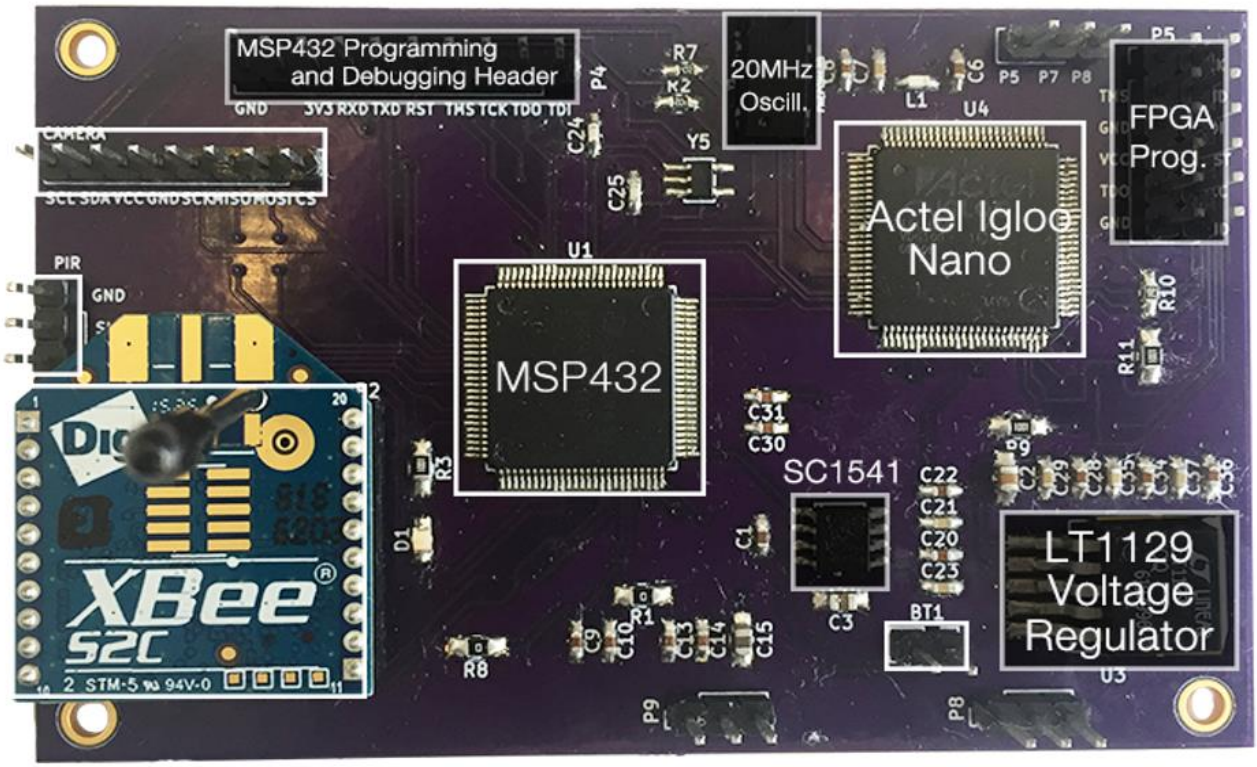

Figure 14 - Physical PCB with Labels for Different Components

3. Experimentation

3.1 Experiments Performed

The main experiments performed in our project were related to power consumption of the system in different modes of operation. Specifically, we conducted experiments in the following modes of operations: normal, image capture. The normal mode of operation is defined as the operation of the MSP432 in LPM3.5 waking up every 30 seconds, waking up the XBee module, staying awake for 2 seconds, if no user request is pending, putting the XBee module back to sleep, and going back to LPM3.5. This operation was measured over a period of 60 seconds, of which the system is awake for 4 seconds. The next mode of operation is when an image is being captured, and subsequently transmitted to the coordinator device. This process was found, on average, to take approximately 15 seconds. A detailed breakdown of the current, power consumption and duty cycle of the different components in these modes of operation can be seen in Table 1 and Table 2 respectively. Additionally, Table 3 shows the expected lifetime of the node, given the data measured in the two aforementioned modes of operation.

3.2 Data

Component	Operation	Current	Vcc	Power(w)	Expected Duty Cycle(min)
XBee S2C	Sleep	0.3μA	3.3V	0.99μW	94%
	Transmit	28mA	3.3V	92.4mW	3%
	Idle	9mA	3.3V	2.97mW	3%
PIR Sensor	Idle	30μA	3.3V	99μW	100%
Camera	Camera Shutdown w/MIC2090	5μA	3.3V	16.5uW	100%
MSP432	LPM3.5	15μA	3.3V	49.5μW	94%
	Active	5mA	3.3V	16.5mW	6%
Voltage Regulators	Active	60μA			100%
Average		1.5mA	3.3V	4.95mW	

Figure 15 - Power Consumption Data Table 1

Component	Operation	Active/On	Vcc	Power(w)	Exp Duty Cycle(15s)
XBee S2C	Transmit	28mA	3.3V	92.4mW	90%
	Sleep	0.6µA	3.3V	1.98µW	10%
PIR Sensor	Motion Detected	80µA	3.3V	264µW	7%
	Idle	30µA	3.3V	99µW	93%
Camera	Active	93mA	3.3V	306.9mW	10%
	Shutdown w/MIC2090	5µA	3.3V	16.5µW	90%
MSP432	Active	5mA	3.3V	16.5mW	100%
Average		39.5mA	3.3V	130.5mW	

Figure 16 - Power Consumption Data Table 1

4. Experiment Validation

To experiment and test our overall project and confirm validation, we broke it down into different sections:

App:

- The individual buttons on the app were tested for functionality and ease of use
- Setup Node: This button opens the QR code app that was linked to our app and scans the code, the QR code contains the node's public key and the address. This information is then successfully sent to Dropbox where it is then read and deleted on the gateway.
- Key Change: This button uploads a file to Dropbox, the file is then retrieved on the gateway and the key changed is executed, we verify this because we print out the new session key on the BBB console.
- Access to Cloud: This button allows the user access to the images on the Dropbox cloud, it is easily validated because we can see the images that were taken with timestamps on the app.
- User Request: Also, like the Key Change button this also uploads a file to Dropbox but with different content inside. The BBB retrieves this data and then sends the command to the node and the image is captured. We verify this because we get an alert on the app that a new image was captured and can see the image within our app.

BeagleBone Black:

- Testing and validation on the BeagleBone was essential since it behaves as a central hub connecting the cloud and app side of our application to the wireless camera sensor node.
- Testing code validation on the BeagleBone
- Confirm and validate commands being sent from App to the node to perform various operations
- Test if our implementation of El-Gamal yielded correct results.
- Validate calibrate image sensor based on the images being received from our Node.

Node:

The testing of microcontroller to FPGA interface was done on the MSP430 and the Xilinx Nexys 4 Board. The microcontroller communicates with the FPGA via I2C communication. This interface of microcontroller to FPGA allows the microcontroller to communicate specifically with the Elliptic Curve Cryptography Core provided to us. The ECC Core is written in VHDL and is inside the FPGA and is to ultimately perform the ECC operation of scalar multiplication needed for our project. Upon successful transmission of data the between microcontroller and FPGA, further testing was moved onto the MSP432 microcontroller and the Actel Igloo nano FPGA, both of which are our chosen microcontroller and FPGA for our project. To confirm that we are able to get correct ECC scalar multiplication outputs back to the microcontroller, we compared them with NIST Test Vectors. Also in order to confirm our full decryption of the session key output we printed the python session key from the Gate way and also the output coming from the FPGA into the console of Code Composer in Figure 23. Fortunately, our results matched.

4.1 Results

4.1.1 Power Consumption

Operation Mode	Current	Vcc	Power(w)	Exp Duty Cycle(day)
Normal	1.5mA	3.3V	4.95mW	99.65%
Capture Image	39.5mA	3.3V	130.35mW	0.35%
Average	1.63mA	3.3V	5.37mW	

Figure 17 - Power Consumption Data Table 2

Battery Life = 0.7∗6300mAh/1.63mA = 2705 hrs. = 112 days

4.1.2 Android App Design

Figure 18 - Main Screen with Four Functions

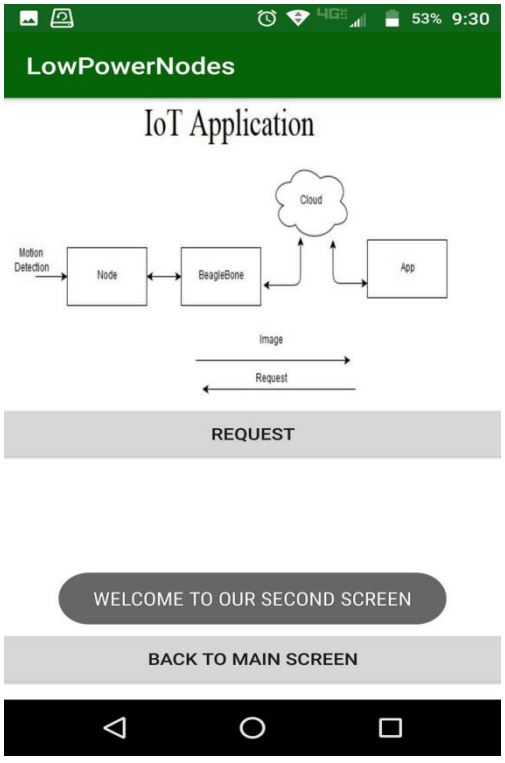

Figure 19 - Second Screen

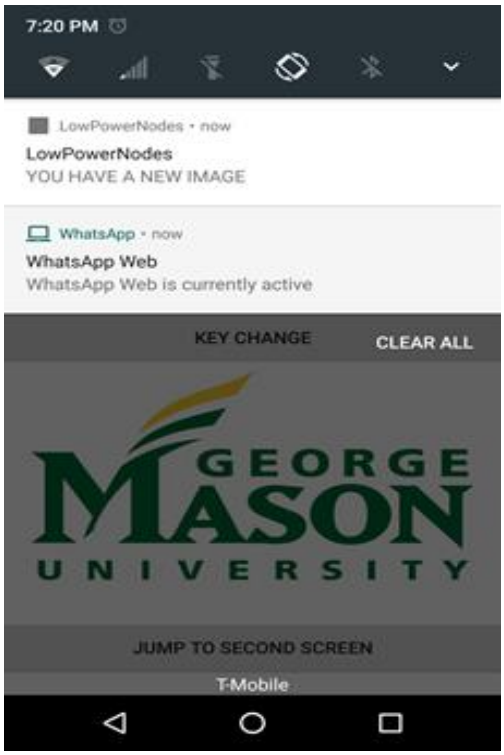

Figure 5 - In-app Notification from Google

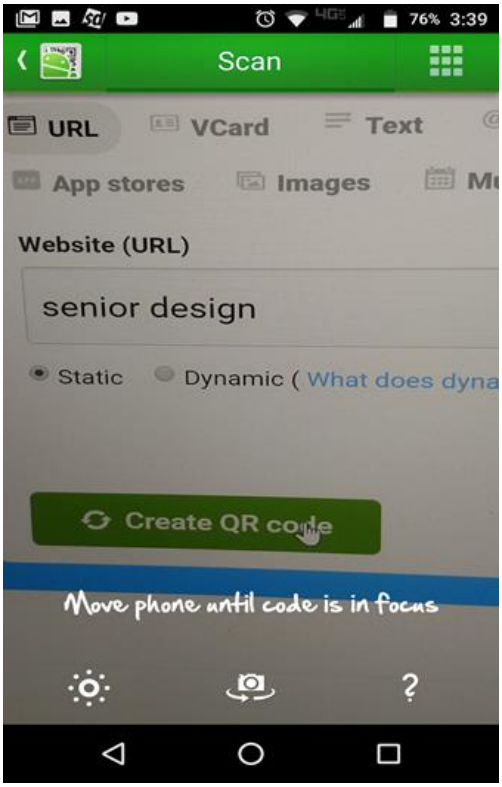

Figure 6 - QR Code Scanner to Obtain Public Key and Address of Node

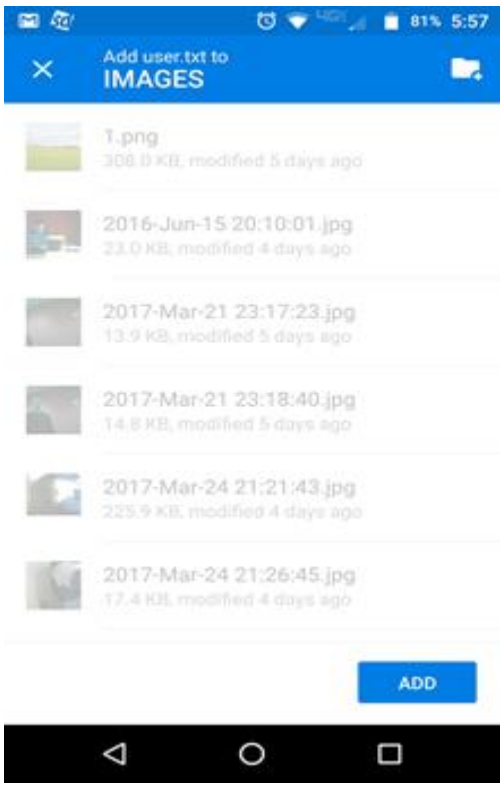

Figure 22 - Image Browser for Images Taken by the Node

4.1.3 Session Key

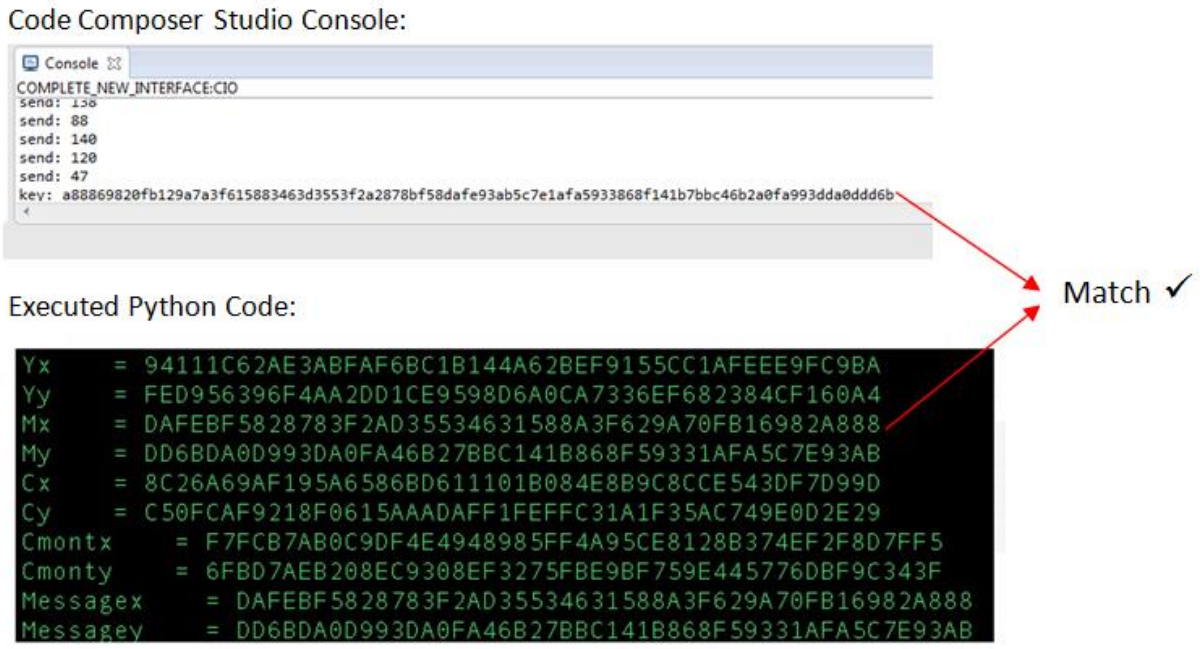

Figure 23 - Session Key Confirmation

We use Code Composer Studio to print out data sent and received from the FPGA. Looking at the session key that is being printed out in Code Composer, we see that the key it is being printed out in little-endian form due to the fact that the ECC core outputs its values in little-endian form. These results show us that we are able to send an encrypted session key from the gateway into the node and properly decrypt the encrypted session key so that both the gateway and node have the session key.

5. Other Issues

a. Reason for the project. Who would benefit, how would they benefit and what would the benefit be? What could be the impact of success? Engineers solve problems to the benefit of humanity.

The purpose of this project was to design a platform from which future IoT devices can be designed with. As we have stated, security was a large concern with IoT devices, so this would benefit users by assuring that their data is secured and privacy is not breached. Additionally, our specific IoT device was also a wireless sensor node, and we designed it in a way that is expandable to many different types of node, and also takes of advantage of low power operation.

b. Potential use of the project. Could the project lead to a marketable product, service or function? Who might the users or purchasers of this product/service/function be? Would the product/service/function serve only local needs or could it be applicable on a wider, potentially global, range of users?

The project itself was not designed so that it can be sold and distributed, rather the design and concept of the project is something that can be implemented into all types of IoT devices and sensor nodes such as smart watches, home security systems, environment monitoring etc.

c. Cost figures for the project. Include parts, rough assembly/coding time/costs, student time spent on design and implementation of the project, and (based on the student time spent) estimated cost of design and this first prototype.

The total cost for parts in this project, as can be seen in section 6.4.1 was $411.41. Time spent on this project which included software design, hardware design, building and testing prototypes was a total of 1645 hours. At a bill amount of $30/hr. the total cost for a first prototype would be $49350.

d. Alternatives to the implemented design that could impact costs and effects on resources use and the environment and community. Alternatives to the chosen design solution.

As we were designing this project, there were other components that were considered but were ultimately decided against. One such component is using a microcontroller that has an integrated ZigBee protocol and has enough processing power for the encryption and decryption. This device would have reduced the cost for components as four separate components are all now integrated into one. However, it would have greatly increased the development cost in terms of man hours and labor, because this chip ran a Real-Time Operating System which none of our team members were familiar with and would take a significant amount of time to learn and implement all our functions with. Additionally, the encryption algorithm that we are currently using is developed in VHDL, and converting this to the microcontroller language would require much analysis and effort.

e. Maintainability/maintenance of the final design solution. (Note, software must be maintained as well as hardware.)

The design as far as software goes is easily adaptable, all the components we used are reprogrammable and can be updated whenever needed, on the PCB we included pins that can allow for easy updates to the microcontroller as well as the FPGA. Also, if the communication protocols need to be changed or updated the communication module can be easily removed as it is not soldered down directly, rather it inserted into the header pins the PCB board which are soldered.

f. Retirement, replacement, or disposal of the project at the end of its "lifetime", i.e. environmental disposal of parts such as batteries, recyclability of parts at time of disposal, documentation needed to allow a replacement or upgrade to be designed.

As our project, does include batteries to power it, proper disposal is required to maintain the integrity of the environment and those around it. Depending on where the product is used, local laws should be followed in accordance to how batteries should disposed of. Our product used 3 AA batteries, and in Virginia batteries such as these, alkaline batteries can be disposed of in household trash or at waste management facilities [7].

6. Administration

6.1 Task Completion

All the functionality for the project that was initially specified is complete. These functions include, motion detection, image capture and image transmission. Additionally, we can successfully communicate with the ECC core that was provided to us by Ahmad Salman. This core is used to decrypt the encrypted session key, which is received from the Gateway.

On the Gateway portion of our project, we were able to receive an image from the node upon motion detection and manually request an image to be taken. Also on the Gateway we can run an El-Gamal Encryption scheme used to encrypt the session key, and we can verify its functionality because after the decryption takes place on the FGPA we acquire the same session key on both sides. Another task that was completed is distinguishing various user requests such as rekeying, setting up a new node, and image requests. The image request is sent over the node and the consequent image is captured and received back on the Gateway. Rekeying, is done similarly, when the request is received, we re-run the encryption scheme and once again both the gateway and node acquire a new identical session key. The setup function is done so that it involves little user involvement besides selection the "set up" button the app and scanning the QR code, which was one of our requirements at the onset of the project.

Finally, all these commands and images are channeled through our Dropbox server and images are uploaded with a time stamp. This data is available for the user to access on the App.

Unfortunately, when it came down to moving our working design into the final PCB, we had errors due to circuit design. Because of this and time constraint we were not able to have all the node functions operate as desired on the PCB, specifically the decryption part of the scheme which when using development kits works as seen in the results section.

6.2 Changes

We did have to change our components a few times due to incompatibility and ease of use that other components provided. This of course did shift our schedule a bit because we had to order the new components and configure them properly. Additionally, what we originally scheduled to have done a few weeks before the end of the due date was the encryption algorithm, however due to various factors such as complexity and effort involved in implementing it in our design we had to shift our full implementation of the encryption to the last week before the end date. And due to this we were not able to finalize our PCB also until about a few weeks before the final date, and were only able to send one version of the PCB out for fabrication rather than being able to send multiple out and make any necessary changes.

6.3 Extra Activities

Other than a few component changes we did not have any unspecified activities to carry out.

6.4 Funds Spent

6.4.1 Development Costs

Item	Quantity	Cost
Actel Igloo nano Starter Kit	1	$100.00
FlashPro4	1	$51.25
MSP430FR5994	1	$25
CC2650 Launchpad	1	$29
MSP432 Launchpad	2	$26.36
OV2640 CMOS Sensor	1	$10.49
Arducam OV2640	1	$25.99
HC-SR501	5	$12.96
XBee S2C	2	$29.99
XBee breakout board	1	10.00
PCB Cost	3	$70
BeagleBone Black	1	$50
	Total:	**$411.04**

Figure 24 - Development Cost

Most of the cost for the project was spent on purchasing the various development boards that were necessary and then also the programming interfaces that were needed for some of them. Additionally, the cost for the PCB was more than expected because we had to expedite the shipping.

6.4.2 Production Costs

Item	Quantity	Cost
AGLN250V2-VQG100	1	$16.11
Arducam OV2640	1	$25.99
HC-SR501 PIR	1	$1.69
XBee S2C	1	$17.50
MSP432P401R	1	$5.29
LT1129	1	$2.92
PCB	1	$7.93
Other Components	1	$5.00
	Total	**$82.43**

Figure 25 - Production Cost

The above table represents the cost of a single node, with the MSP432, FPGA, camera and motion sensor we selected for the project. The prices are from the lowest available options, which for some is when bought in bulk, and the PCB is with ground shipping from the lowest cost manufacturer.

6.5 Man Hours

Team Member	Man Hours
Irfan Malik	250
Logan Robson	300
Ishak Nur	280
Jesse Esquivel	280
Mohammed Marjan	285
Hung Tran	250
Total Hours:	**1645**

Figure 26 - Man Hours

Most time spent on this project was on writing code for the various components we were using in our project. These include the FPGA, MSP430 for testing, MSP432, the BeagleBone Black as our gateway and camera module. Additionally, since we were all new to PCB design, we needed to learn how to first design a PCB on KiCad and then understand how the layout should be on the actual PCB board, this in conjunction with finding and selecting all the necessary components such as voltage regulators and oscillators took a significant amount of time as well. Finally, since we wanted to create an understandable and easy user interface, a good portion of time was spent on the developing the app, and allowing all the buttons to carry out a specific function, such as requesting an image, or setting up a new node.

7. Lessons Learned

7.1 Knowledge and Skills Learned

We learned a lot of lessons including teamwork, design approach, design process, and research. Research has been vital throughout our whole project. Also, we learned how document an effective proposal, progress reports, and other important information related to our project. Learning to ask for help has been another major skill we gained. Early on it was clear we needed a lot help, and we had to learn to communicate effectively with our faculty advisor and a graduate student who are experts in the topics included in our project.

Among the technical skills we acquired from this project, the most prominent are in embedded system concepts and design. Specifically, over the course of this project we have become much more adept at using and developing with, serial protocols. Mainly the I2C, SPI and UART protocols, in interfacing with the XBee module, Arducam and FPGA. Additionally, we gained experience working with TI's Driver Library, which is used in a number of TI's microcontrollers.

Another technical skill acquired from this project is the use of KiCad to go from a circuit schematic to a PCB. This process entailed learning about PCB layout, PCB footprints, as well as scouring datasheets to find parts that would work in our design. One of the biggest adjustments, when transitioning from the circuit schematic of our design to a PCB layout, was attempting to visualize where each of the components on the PCB should lie. For this process, we found it best to first hand draw the ideal layout of the PCB, given the connections in the circuit schematic. For example, the Arducam OV2640 had the majority of its connections terminating at ports on the left side of the MSP432 (with Pin 1 oriented to the NW), hence the placement of the Arducam header pins should be to the left of the MSP432. Another challenge was finding components necessary for our design, available in hand-solderable footprint. This was mainly a challenge when choosing oscillators for the MSP432 and FPGA.

7.2 Team Experience

Working throughout two semesters with a team to develop a challenging project helped us gain a lot of valuable lessons and information. We've learned a lot from each other, all of us have something we can contribute to the team and it is necessary to understand each other's capabilities in order to allocate tasks appropriately. What one of us may struggle with another may excel at, and if one of us are stuck on a problem having another set of eyes with a different perspective can completely change the outcome and possibly save a tremendous amount of time. Additionally, we've learned to not take anything personally, if one of us is doing something incorrectly, and someone points out the issue, ultimately it is for the benefit of the team so that we can solve the issue and continue, it's not to insult anyone's work or to hinder them from continuing.

References

[1] http://meroli.web.cern.ch/meroli/lecture_cmos_vs_ccd_pixel_sensor.html
[2] http://www.arducam.com/arducam-mini-released/
[3] http://vip.sugovica.hu/Sardi/kepnezo/JPEG%20File%20Layout%20and%20Format.htm
[4] http://qyx.krtko.org/projects/ov2640_stm32/
[5] https://embsi.blogspot.com/2013/01/how-to-use-cellular-ram-from-micron.html
[6] https://github.com/oetr/FPGA-I2C-Slave
[7] http://www.wikihow.com/Dispose-of-Batteries
[8] www.digikey.com/product-detail/en/microsemi-corporation/AGLN-NANO-KIT/1100-1136-ND/2745083
[9] https://www.controlanything.com/Relay/Device/A3001
[10] https://blogspot.tenettech.com/configuring-xbee-in-sleep-mode.html
[11] http://www.libelium.com/development/waspmote/documentation/x-ctu-tutorial/
[12] https://github.com/wobine/blackboard101/blob/master/EllipticCurvesPart4-PrivateKeyToPublicKey.py
[13] khoazend, "Khoa Pham," YouTube.https://www.youtube.com/user/khoazend.
[14] Is Dropbox safe to use? - Dropbox Help," Dropbox .https://www.dropbox.com/en/help/27
[15] Jens-Peter Kaps. ECE 746: Lecture 3 Elliptic Curve Cryptosystems. George Mason University
[16] Ahmad Salman. Member of Cryptographic Engineering Research Group(CERG).George Mason University

Appendix A: Proposal

Appendix B: Design Document