Program: ESE 4009

INSTRUCTOR: Prof. Mike Aleshams

Group # 8

Student Name	Student ID	Signature*
Zain Rajani	c0752681	Zain Rajani

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties.*

Project Proposal

Project Title: I-LETTERBOX (Intelligent Letter Box)

Description of the latest similar system:

SMART LETTERBOX USING ARDUINO AND GSM

Letters are an important part of daily business. People receive letters either from the government offices for either tax payments, or approval of their loans and schemes etc. Thus people at times forget to check their mailboxes and some days people don't get the mails at all. Thus, a system was needed where when any letter arrived for the user he/she must be notified or must be made aware of through some system. Therefore such a prototyping system was created of which we give a short description.

First, let us look at the block diagram for the existing system.

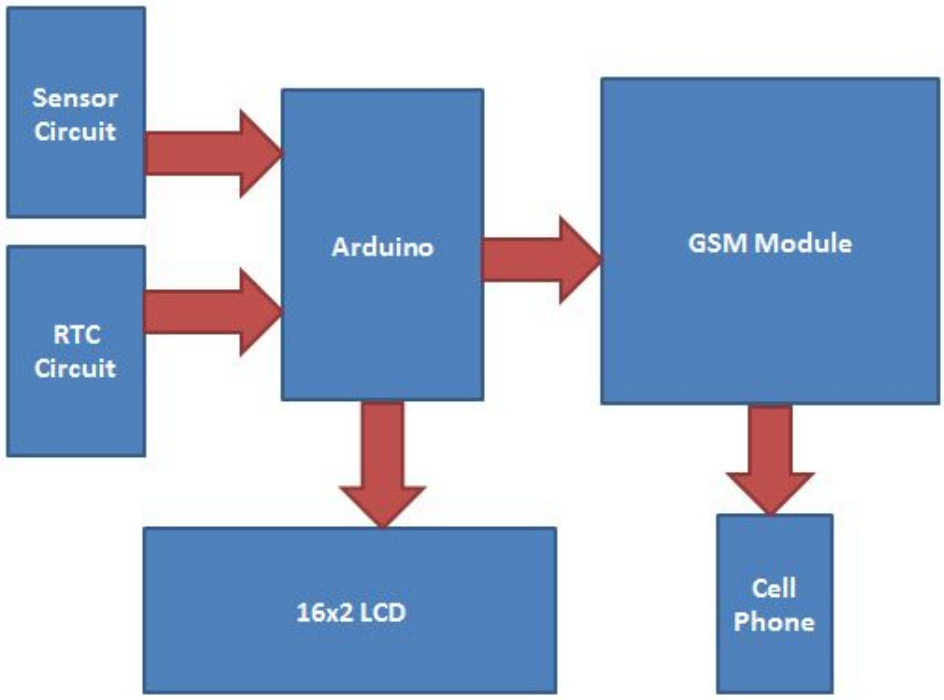


Figure 1: Existing System Block Diagram

Let's describe the process of how this system works. First, when the system is powered up all the sensor networks, GSM Module, RTC (Real Time Clock), LCD Display gets activated and start to perform their respective task. Here in the sensor circuit, the IR Transmitter Reciever is used. This is used to detect if the letter arrives in the box or no. Once any letter is detected the LCD receives an update displaying the letter has been arrived and along with the letter number and the date and time of arrival. The date and time that gets displayed are through the RTC Circuit which stores the value in EEPROM. Along with the LCD display getting updated each time the GSM module also sends an SMS (Short Message Service) to the registered mobile number intimating the user that a letter is deposited in the box.

Limitations of the latest similar system:

Though the system is of utmost use; the system possesses certain limitations. Work has been carried further by just replacing some of the sensors just like replacing the IR sensor by PIR sensor and instead of using a power supply. Thus, we list some of the limitations of the system as follows:

- The use of Arduino which is an 8-bit microcontroller thus the processing speed is slow. Though one cannot deny that it has an easy to use development network.
- Though the system maintains the count about how many letters have been delivered it doesn't decrement the count when the letters have been removed from the box.
- Since the modern era is using the cloud and Internet of Things (IoT) as the latest technology this system lacks it.
- As some letters may be of no importance and some may be of great importance this system doesn't have any system/ button feature which can intimate the user that there is an urgent mail in the letterbox.

Solution 1:

✔ **Block Diagram**

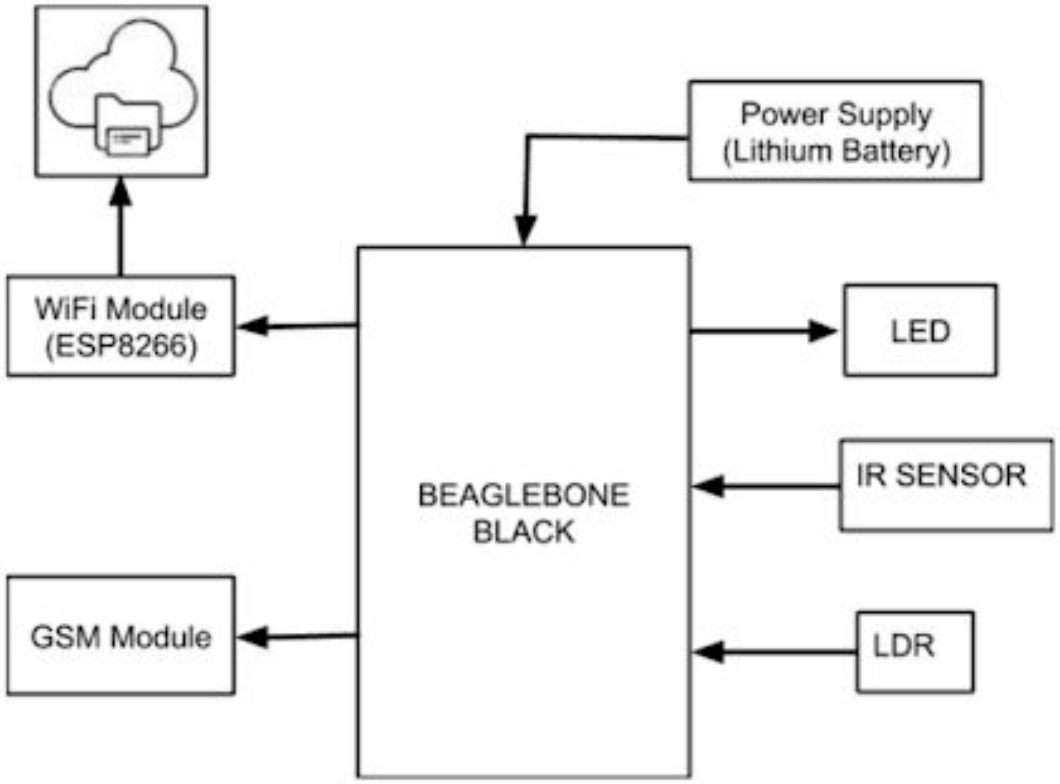
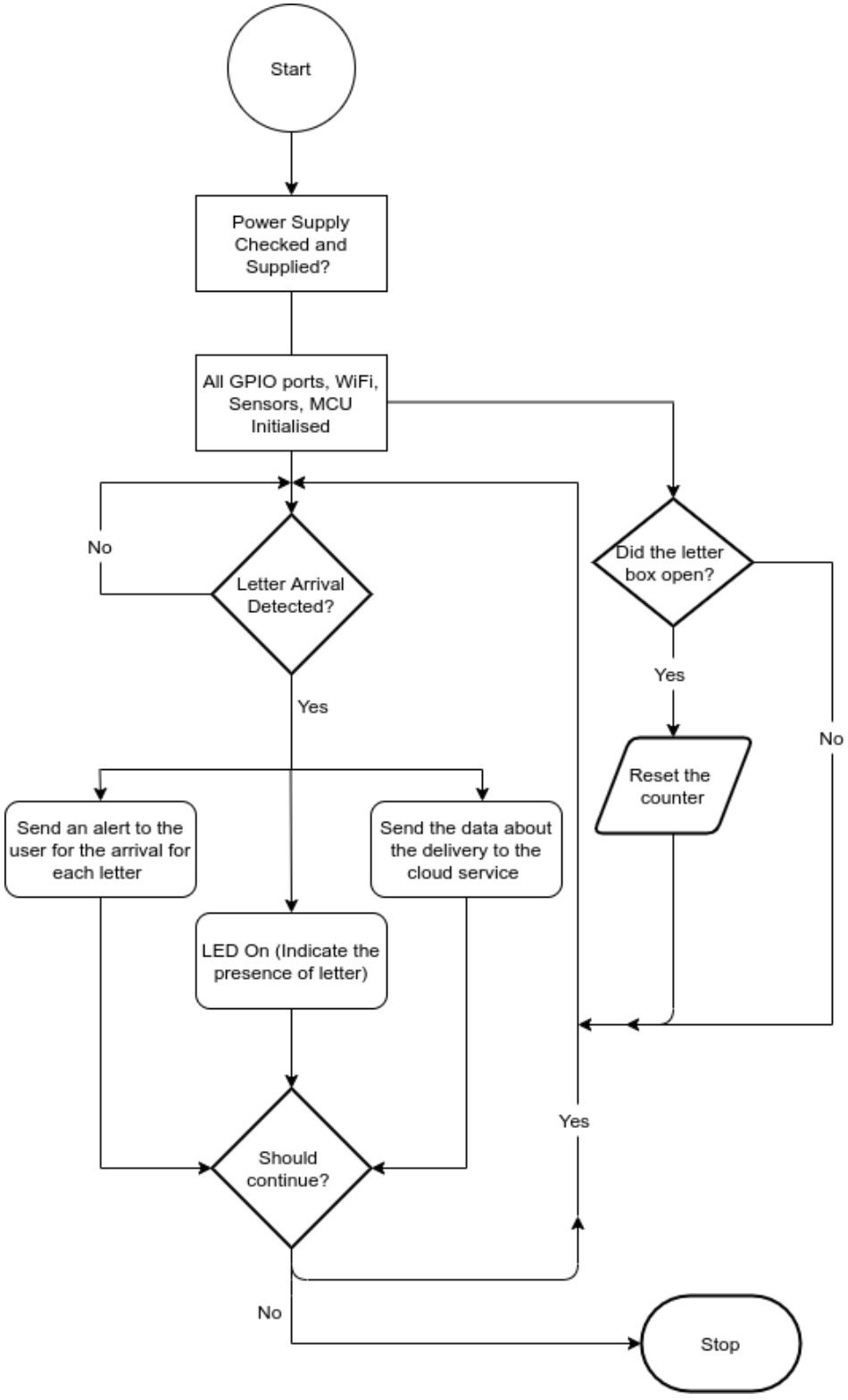


Figure 2: Proposed Overview of Solution 1

Looking at the limitations we could suggest that we can replace the Arduino microcontroller by a 32-bit Processing device such as the **BeagleBone Black**. The system can use the same **IR sensor** to detect the arrival of the letter in the box and this output can be used to increment the counter. Here we think that we will be having a small gap from where the letters will come. Also, once the letters have been removed then the software counter should be reset. This can be done with the help of an **LDR (Light Dependent Resistor)**. Also once the letter has arrived a texting service through GSM Module or using some third party applications such as Twilio or IFTTT can be used. The Cloud service can be used to upload the letters arrived in the box. Since the Beaglebone Black doesn't have inbuilt Wifi thus we can use a **Wifi Adapter** through which a cloud communication can be established. The LED attached can be used to indicate that a letter is present in the box.

FLOWCHART:


Start
Power Supply Checked and Supplied?
All GPIO ports, WiFi, Sensors, MCU Initialised
Letter Arrival Detected?
No
Yes
Did the letter box open?
Yes
No
Reset the counter
Send an alert to the user for the arrival for each letter
Send the data about the delivery to the cloud service
LED On (Indicate the presence of letter)
Should continue?
Yes
No
Stop

✔ **Features**

- Uses BeagleBone as the main processing
- IR Sensor is used to check the arrival of the letters along with LDR which will clear the count once all letters have been removed from the box
- Communication with the cloud and via GSM for user intimation about the details of the arrival and arrival message respectively.

➢ **Use of various peripherals such as touch screens, cameras, microphones and speakers, GPIOs, timers, GPS modules, Bluetooth, WiFi, and ADC/DACs?**

- The system uses the GPIOs (General Purpose Input Output) pins to communicate with the sensors that are been connected to the device (BeagleBone)
- IR Sensor and LDR will be used as a combination to check the arrival of the letter and indicate if the letter is removed respectively.
- BeagleBone will be connected to the cloud through the WiFi module like the ESP8266

➢ **Use of I2C, SPI, RS232/RS-485, IrDA infrared, JTAG, USB, Bluetooth, IEEE 802.11 WiFi, IEEE 802.3 Ethernet, CAN and GPS protocols and systems?**

- Digital/ Serial Communication: All the devices connected on the Beaglebone would use the Digital Communication to send the data to the BeagleBone device
- For sending the data to the cloud we use IEEE 802.11 WiFi protocol for successful cloud communication
- ThingsSpeak Cloud communication platform will be used to update the details in a timely manner.
- The connection to the Beaglebone device shall be through SSH (Secure Shell) which uses the TCP (Transfer Communication Protocol)

➢ **Use of preemptive versus cooperative scheduler operation; tick rate and time slicing; critical code; fixed, dynamic and hybrid task priority allocation; application-specific considerations; power management tactics; semaphores, mutexes and queues; debugging strategies; performance estimation?**

- Since one of the aim of the project would be to reduce power consumption thus preemptive scheduler can be used.

✔ Hardware and Software Requirement

Hardware Requirements:

- BeagleBone Black (BBB): Used as the main processing unit a.k.a the master device. All the sensors and devices shall be connected to this device
- IR Sensor (Obstacle Sensor)
- LDR (Light Dependent Resistor)
- ESP8266 (WiFi Module)
- GSM Module (SIM300)
- Power Supply (Lithium Battery)
- LED (Light Emitting Diode)
- Connecting Wires
- Jumper Wires
- Basic Electronic Components: Multimeter (to check on voltages and current at various nodes basically for debugging the circuit), Wire Cutters (Cut off the wires of the required lengths to make the circuit look clean), Strippers (To remove the insulation and connect)
- Laptop Device (Equipped with appropriate storage and capable to handle the BeagleBone having a Linux OS preferrably)
- BreadBoard (Used to make the circuit for testing before final PCB design and implementation)
- Soldering Equipments (Soldering Iron: Solder the components on the PCB, Desoldering Gun: Removes the excess of soldering or if soldered incorrectly, Soldering Wire: used as electrical conducting to join the connection)
- SD Card: To store the image and the data related to the Beaglebone

Software Requirements:

- Linux OS (Debian): Flashed on the Beaglebone with the latest version. Usually Beaglebone devices come with pre-installed images but sometimes may require updating.
- GCC (GNU (GNUs Not Unix) Compiler Collection): Usually pre-installed on the linux distribution. It would be used to compile the code and generate the output file.

- Eclipse: Some of the programming for the hardware would be done using this software and the cross compiler may be used and the file may be sent on the beaglebone for execution.
- Nano (Editor): Any coding if required to be done in Beaglebone itself thus, a nano editor will be used to write the program and save the file using appropriate extension and this file shall be compiled using the GCC to get the binary executable file.
- ThingSpeak: Open-Source Internet of Things (IoT) application and API used to store and retrieve data using the HTTP (Hypertext Transfer Protocol) and MQTT (Message Queuing Telemetry Transport). It works closely with MATLAB and MathWorks. For the project purpose we will use it as a cloud platform to transfer the data and store it in the cloud in real time.
- EasyEDA: A PCB design and simulation tool enabling hardware engineers to write EDA (Electronic Design Automation). We use this tool for PCB design and layout. It could be used as simulation but beaglebone simulation may not be possible.
- Programming Language Support: For the project purpose we will try to program majority in either C/ C++ but we may use python in situation where certain libraries may not be present in C/C++

✔ References:

Khan, S. (2015, September 23). Intelligent Letter Box using Arduino and GSM. Retrieved September 26, 2020, from https://www.engineersgarage.com/contributions/intelligent-letter-box-using-arduino-and-gsm/

Electronic Letter Box: Embedded Systems Project Topics. (n.d.). Retrieved September 26, 2020, from https://www.seminarsonly.com/Engineering-Projects/Embedded/Electronic-Letter-Box.php

Devi Pujari, A., Bansode, P., Girme, P., Mohite, H., & Pande, A. (2016). Smart Letter Box System Using Obstacle Sensor For Notifies The User By Android Application. *International Research Journal of Engineering and Technology (IRJET), 3*(10), 823-825.

Raithatha, D. (2017, October 02). Smart Letter Box. Retrieved September 26, 2020, from https://www.instructables.com/id/Smart-Letter-Box/

Electronic Letter Box Project Circuit and its Working. (2018, May 26). Retrieved September 26, 2020, from https://www.electronicshub.org/electronic-letter-box-project-circuit/

Solution 2:

✔ **Block Diagram**

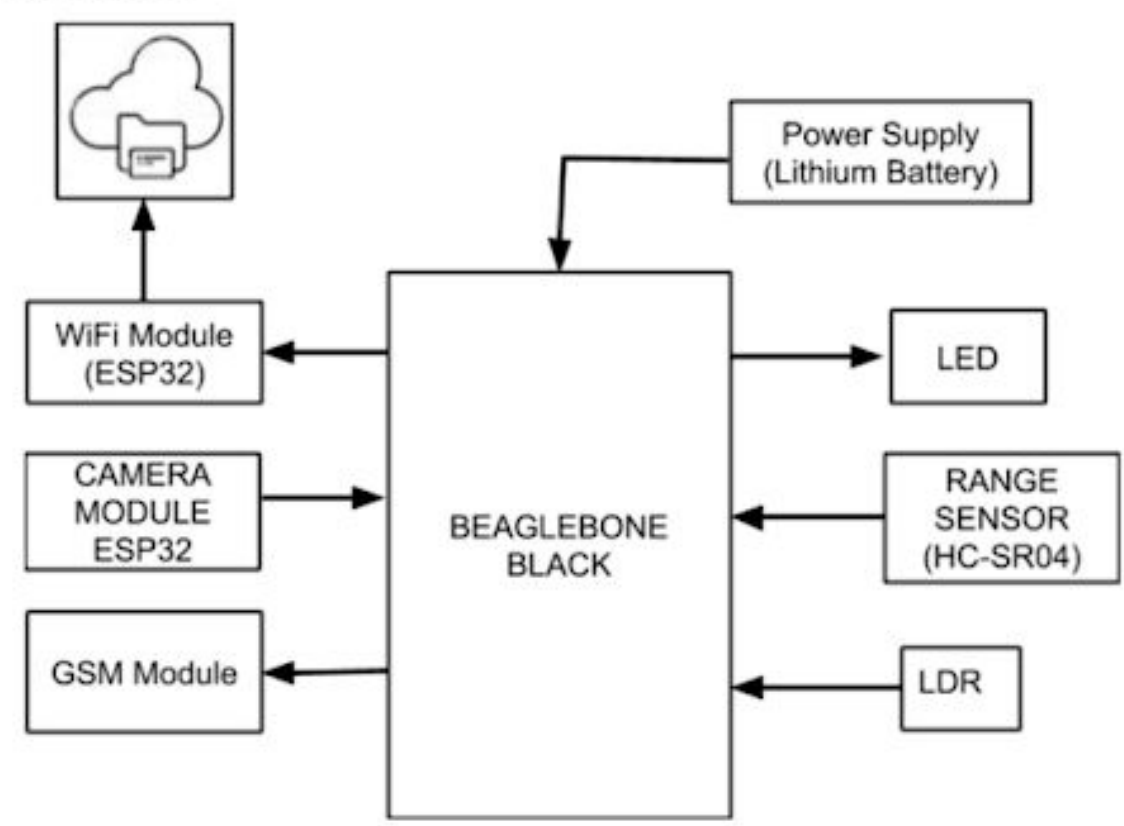
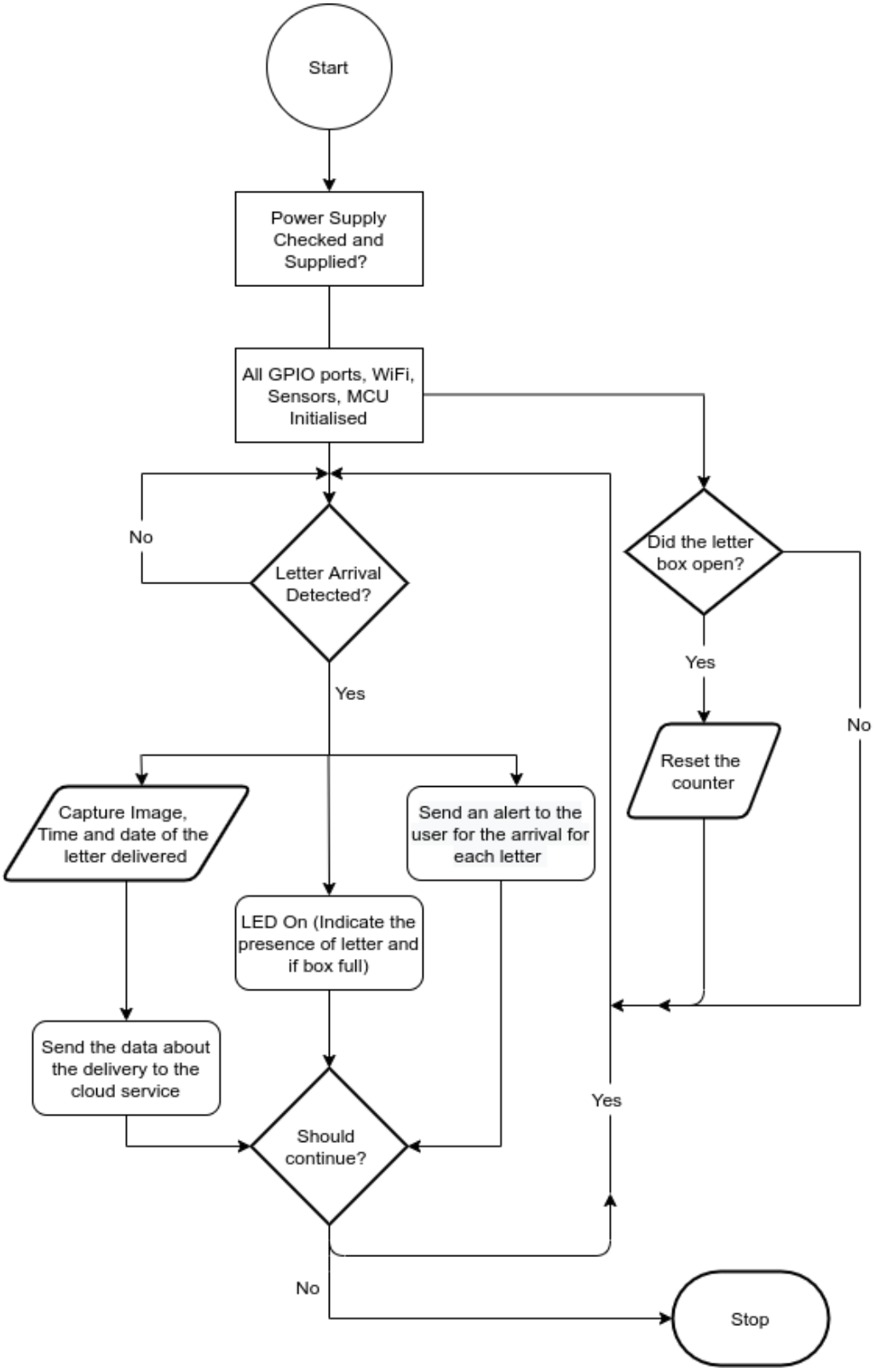


Figure 3: Proposed Overview of Solution 2

This proposed solution uses the replacement of IR sensor by **Ultrasonic sensor** assuming the fact that the letter box has slits made where the letters are to be inserted thus as the letters are inserted the distance from the sensor to the end wall of the box will change and this can be used to detect the number of letters assuming that the slits are made at fixed distance. Along with this the letter box will be equipped with a **camera module** that can be used to capture the picture as to who deposited the mail in the box. Also when the distance changes i.e. when a letter is deposited in the box the software counter will increment and send a text message to the user via **GSM** and also the date and time of the letters along with the image will be sent to the **cloud platform**. The **LDR** in this case will be used to reset the counter with certain limitations. Also once all the slits are full the LED can be used as an indicator. A separate **LED** if required may be attached to the system to indicate in case we have a single letter also present in the box or if the box is full of letters.

FLOWCHART:


Start
Power Supply Checked and Supplied?
All GPIO ports, WiFi, Sensors, MCU Initialised
No
Letter Arrival Detected?
Yes
Capture Image, Time and date of the letter delivered
LED On (Indicate the presence of letter and if box full)
Send an alert to the user for the arrival for each letter
Send the data about the delivery to the cloud service
Should continue?
Yes
No
Stop
Did the letter box open?
Yes
Reset the counter
No

✔ **Features**

- Uses BeagleBone as the main processing unit which is 32 bit
- To check if the letter is arrived we use the range sensor (HC-SR04) which calculates the distance between the sensor and the wall/ last letter deposited in the box and this can be used to calculate the number of letters
- Communication with the cloud and via GSM for user intimation about the details of the arrival and arrival message respectively.
- Capturing the details with respect to the letter like either the image of the letter or person depositing the image can be seen using the camera attached.

➢ **Use of various peripherals such as touch screens, cameras, microphones and speakers, GPIOs, timers, GPS modules, Bluetooth, WiFi, and ADC/DACs?**

- The system uses the GPIOs (General Purpose Input Output) pins to communicate with the sensors that are been connected to the device (BeagleBone)
- Range Sensor and LDR will be used as a combination to check the arrival of the letter and indicate if the letter is removed respectively.
- BeagleBone will be connected to the cloud through the WiFi module like the ESP32
- The system is powered up using the lithium battery as the system is outside but stationary.

➢ **Use of I2C, SPI, RS232/RS-485, IrDA infrared, JTAG, USB, Bluetooth, IEEE 802.11 WiFi, IEEE 802.3 Ethernet, CAN and GPS protocols and systems?**

- Digital/ Serial Communication: All the devices connected on the Beaglebone would use the Digital Communication to send the data to the BeagleBone device
- For sending the data to the cloud we use IEEE 802.11 WiFi protocol for successful cloud communication via ESP32 which is the successor of ESP8266
- ThingsSpeak Cloud communication platform will be used to update the details in a timely manner.
- The connection to the Beaglebone device shall be through SSH (Secure Shell) which uses the TCP (Transfer Communication Protocol)

➢ **Use of preemptive versus cooperative scheduler operation; tick rate and time slicing; critical code; fixed, dynamic and hybrid task priority allocation; application-specific considerations; power management tactics; semaphores, mutexes and queues; debugging strategies; performance estimation?**

- Since one of the aim of the project would be to reduce power consumption thus preemptive scheduler can be used.

✔ **Hardware and Software Requirement**

Hardware Requirements:

- BeagleBone Black (BBB): Used as the main processing unit a.k.a the master device. All the sensors and devices shall be connected to this device
- Range Sensor (Ultrasonic Sensor HC-SR04)
- LDR (Light Dependent Resistor)
- ESP32 (WiFi & Camera Module)
- Power Supply (Lithium Battery)
- GSM Module (SIM300)
- LEDs (x2)
- Connecting Wires
- Jumper Wires
- Basic Electronic Components: Multimeter (to check on voltages and current at various nodes basically for debugging the circuit), Wire Cutters (Cut off the wires of the required lengths to make the circuit look clean), Strippers (To remove the insulation and connect)
- Laptop Device (Equipped with appropriate storage and capable to handle the BeagleBone having a Linux OS preferrably)
- BreadBoard (Used to make the circuit for testing before final PCB design and implementation)
- Soldering Equipments (Soldering Iron: Solder the components on the PCB, Desoldering Gun: Removes the excess of soldering or if soldered incorrectly, Soldering Wire: used as electrical conducting to join the connection)
- SD Card: To store the image and the data related to the Beaglebone

Software Requirements:

- Linux OS (Debian): Flashed on the Beaglebone with the latest version. Usually Beaglebone devices come with pre-installed images but sometimes may require updating.
- GCC (GNU (GNUs Not Unix) Compiler Collection): Usually pre-installed on the linux distribution. It would be used to compile the code and generate the output file.
- Eclipse: Some of the programming for the hardware would be done using this software and the cross compiler may be used and the file may be sent on the beaglebone for execution.
- Nano (Editor): Any coding if required to be done in Beaglebone itself thus, a nano editor will be used to write the program and save the file using appropriate extension and this file shall be compiled using the GCC to get the binary executable file.
- ThingSpeak: Open-Source Internet of Things (IoT) application and API used to store and retrieve data using the HTTP (Hypertext Transfer Protocol) and MQTT (Message Queuing Telemetry Transport). It works closely with MATLAB and MathWorks. For the project purpose we will use it as a cloud platform to transfer the data and store it in the cloud in real time.
- EasyEDA: A PCB design and simulation tool enabling hardware engineers to write EDA (Electronic Design Automation). We use this tool for PCB design and layout. It could be used as simulation but beaglebone simulation may not be possible.
- Programming Language Support: For the project purpose we will try to program majority in either C/ C++ but we may use python in situation where certain libraries may not be present in C/C++

✔ **References:**

Khan, S. (2015, September 23). Intelligent Letter Box using Arduino and GSM. Retrieved September 26, 2020, from https://www.engineersgarage.com/contributions/intelligent-letter-box-using-arduino-and-gsm/

Electronic Letter Box: Embedded Systems Project Topics. (n.d.). Retrieved September 26, 2020, from https://www.seminarsonly.com/Engineering-Projects/Embedded/Electronic-Letter-Box.php

Devi Pujari, A., Bansode, P., Girme, P., Mohite, H., & Pande, A. (2016). Smart Letter Box System Using Obstacle Sensor For Notifies The User By Android Application. *International Research Journal of Engineering and Technology (IRJET), 3*(10), 823-825.

Raithatha, D. (2017, October 02). Smart Letter Box. Retrieved September 26, 2020, from https://www.instructables.com/id/Smart-Letter-Box/

Electronic Letter Box Project Circuit and its Working. (2018, May 26). Retrieved September 26, 2020, from https://www.electronicshub.org/electronic-letter-box-project-circuit/

BeagleBone Black. (n.d.). Retrieved September 28, 2020, from https://beagleboard.org/black

Burnett, R. (2020, July 15). Understanding How Ultrasonic Sensors Work. Retrieved September 28, 2020, from https://www.maxbotix.com/articles/how-ultrasonic-sensors-work.htm

Light Dependent Resistor Circuit Diagram with Applications. (2019, November 11). Retrieved September 28, 2020, from https://www.elprocus.com/ldr-light-dependent-resistor-circuit-and-working/

ESP32-CAM Video Streaming and Face Recognition with Arduino IDE. (2020, April 30). Retrieved September 28, 2020, from https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/

Final Solution (after presentation):

✔ **Block Diagram**

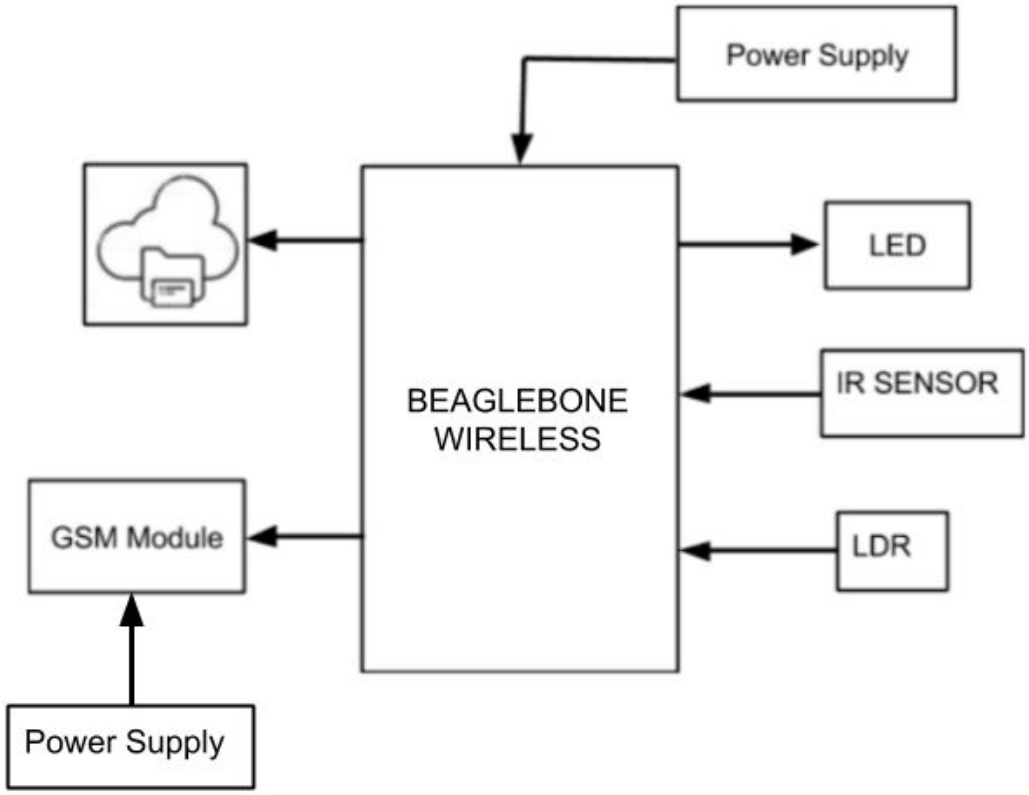
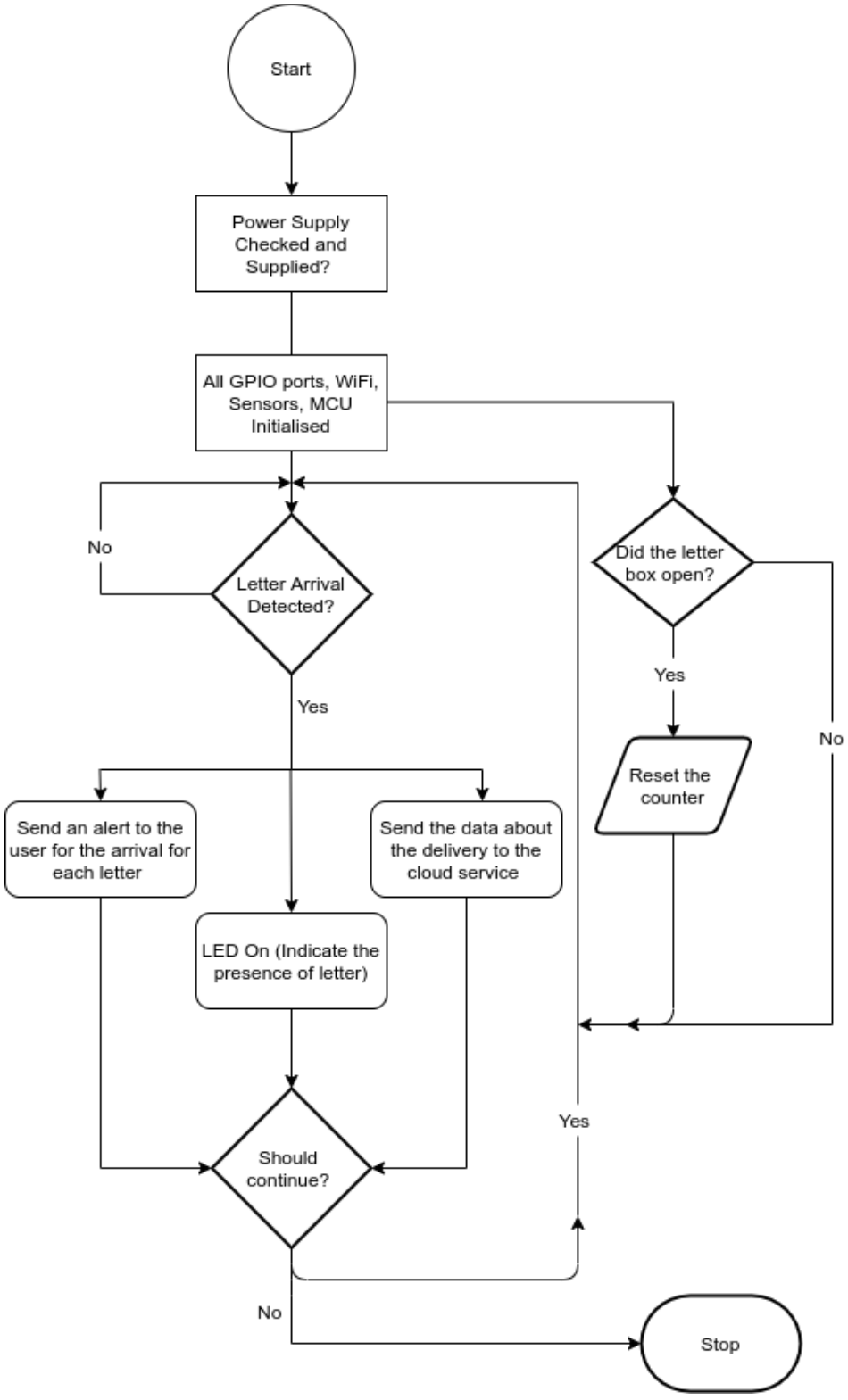


Figure 4: Final Proposed Block Diagram

Looking at the limitations we could suggest that we can replace the Arduino microcontroller by a 64-bit Processing device such as the **BeagleBone Wireless**. The system can use the same **IR sensor** to detect the arrival of the letter in the box and this output can be used to increment the counter. Here we think that we will be having a small gap from where the letters will come. Also, once the letters have been removed then the software counter should be reset. This can be done with the help of an **LDR (Light Dependent Resistor)**. Also once the letter has arrived a texting service through **GSM Module** or using some third party applications such as Twilio or IFTTT can be used. The Cloud service can be used to upload the letters arrived in the box. Since the Beaglebone Black doesn't have inbuilt Wifi thus we can use a **Wifi Adapter** through which a cloud communication can be established. The **LED** attached can be used to indicate that a letter is present in the box.

FLOWCHART:


Start
Power Supply Checked and Supplied?
All GPIO ports, WiFi, Sensors, MCU Initialised
Letter Arrival Detected?
No
Yes
Did the letter box open?
Yes
No
Reset the counter
Send an alert to the user for the arrival for each letter
Send the data about the delivery to the cloud service
LED On (Indicate the presence of letter)
Should continue?
Yes
No
Stop

✔ **Features**

- Uses BeagleBone as the main processing which is a 32 bit processing unit.
- IR Sensor is used to check the arrival of the letters along with LDR which will clear the count once all letters have been removed from the box
- Communication with the cloud and via GSM for user intimation about the details of the arrival and arrival message respectively.

➢ **Use of various peripherals such as touch screens, cameras, microphones and speakers, GPIOs, timers, GPS modules, Bluetooth, WiFi, and ADC/DACs?**

- The system uses the GPIOs (General Purpose Input Output) pins to communicate with the sensors that are been connected to the device (BeagleBone)
- IR Sensor and LDR will be used as a combination to check the arrival of the letter and indicate if the letter is removed respectively.
- BeagleBone will be connected to the cloud through the WiFi module inbuilt on the Beaglebone Wireless

➢ **Use of I2C, SPI, RS232/RS-485, IrDA infrared, JTAG, USB, Bluetooth, IEEE 802.11 WiFi, IEEE 802.3 Ethernet, CAN and GPS protocols and systems?**

- Digital/ Serial Communication: All the devices connected on the Beaglebone would use the Digital Communication to send the data to the BeagleBone device
- For sending the data to the cloud we use IEEE 802.11 WiFi protocol for successful cloud communication
- ThingsSpeak Cloud communication platform will be used to update the details in a timely manner via the MQTT (Message Queuing Telemetry Transport) service.
- The connection to the Beaglebone device shall be through SSH (Secure Shell) which uses the TCP (Transfer Communication Protocol)
- The GSM (Global System for Mobile Communication) will be connected to the Beaglebone board via serial UART

➢ **Use of preemptive versus cooperative scheduler operation; tick rate and time slicing; critical code; fixed, dynamic and hybrid task priority allocation; application-specific**

considerations; power management tactics; semaphores, mutexes and queues; debugging strategies; performance estimation?

- Since one of the aim of the project would be to reduce power consumption thus preemptive scheduler can be used.
- Also, the Beaglebone boards are set to perform the execution through the pre-emptive scheduler and thus this will help in power management through functions like sleep which helps us to achieve task of the pre-emption
- As most of the data is sensed and processed using real time and also the system sends data to the cloud in this manner the system uses real time systems.

✔ Hardware and Software Requirement

Hardware Requirements:

- BeagleBone Wireless (BB-WI): Used as the main processing unit a.k.a the master device. All the sensors and devices shall be connected to this device
- IR Sensor (Obstacle Sensor)
- LDR (Light Dependent Resistor)
- GSM Module (SIM900)
- Power Supply (Wall Adapter as no support for connecting external Li-ion battery)
- LED (Light Emitting Diode)
- Connecting Wires
- Jumper Wires
- Basic Electronic Components: Multimeter (to check on voltages and current at various nodes basically for debugging the circuit), Wire Cutters (Cut off the wires of the required lengths to make the circuit look clean), Strippers (To remove the insulation and connect)
- Laptop Device (Equipped with appropriate storage and capable to handle the BeagleBone having a Linux OS preferrably)
- BreadBoard (Used to make the circuit for testing before final PCB design and implementation)
- Soldering Equipments (Soldering Iron: Solder the components on the PCB, Desoldering Gun: Removes the excess of soldering or if soldered incorrectly, Soldering Wire: used as electrical conducting to join the connection)

- SD Card: To store the image and the data related to the Beaglebone

Software Requirements:

- Linux OS (Debian): Flashed on the Beaglebone with the latest version. Usually Beaglebone devices come with pre-installed images but sometimes may require updating.
- GCC (GNU (GNUs Not Unix) Compiler Collection): Usually pre-installed on the linux distribution. It would be used to compile the code and generate the output file.
- Eclipse: Some of the programming for the hardware would be done using this software and the cross compiler may be used and the file may be sent on the beaglebone for execution.
- Nano (Editor): Any coding if required to be done in Beaglebone itself thus, a nano editor will be used to write the program and save the file using appropriate extension and this file shall be compiled using the GCC to get the binary executable file.
- ThingSpeak: Open-Source Internet of Things (IoT) application and API used to store and retrieve data using the HTTP (Hypertext Transfer Protocol) and MQTT (Message Queuing Telemetry Transport). It works closely with MATLAB and MathWorks. For the project purpose we will use it as a cloud platform to transfer the data and store it in the cloud in real time.
- EasyEDA: A PCB design and simulation tool enabling hardware engineers to write EDA (Electronic Design Automation). We use this tool for PCB design and layout. It could be used as simulation but beaglebone simulation may not be possible as SPICE libraries are currently not present in the software. Hence software simulation of the Beaglebone circuits cannot be done but the PCB can be designed using this software.
- Programming Language Support: For the project purpose we will try to program majority in either C/ C++ but we may use python in situation where certain libraries may not be present in C/C++

✔ **Project Cost Estimation**

COST ESTIMATION FOR I-LETETRBOX PROJECT				
Component Name	**Qty**	**Cost**	**Est. Cost**	**Reference Link**
BeagleBone Wireless	1	$ 113.79	$ 113.79	BeagleBone Black Wireless Buy
LED	2	$ 0.85	$ 1.70	LED Buy
GSM Module (SIM900)	1	$ 27.99	$ 27.99	GSM SIM900 BUY
Cloud Service (ThingSpeak) [Free for 4 Channel extra channel $ 79.00/ year		$ 0.00	$ 0.00	ThingSpeak Price for Students
Light Dependent Resistor (LDR)	1	$ 5.16	$ 5.16	LDR Module Buy Amazon
IR (infrared Sensor)	1	$ 4.99	$ 4.99	IR Sensor Pair
Wall Adapter Barrel Pin	2	$ 11.38	$ 22.76	Type C Wall Adapter
PCB Design and Soldering Equipments		$ 60.00	$ 60.00	Approx
Basic Electronics Kit (Cutters, Wires, Multimeters, etc)	1	$ 75.00	$ 75.00	Soldering Kit Basic
Resistor and Capacitor Kit	1	$ 25.98	$ 25.98	Resistor Kit - Elmwood Capacitor Kit - Elmwood
Net Total			$ 337.37	
Tax (13%)			$ 43.86	
Total			$ 381.23	

✔ **Milestones (Deliverables and Time Schedule)**

Task Name	**Start Date**	**End Date**	**Person In-charge**
Project Proposal	September 18, 2020	October 09, 2020	Zain Rajani
Finalizing Hardware Requirements	October 10, 2020	October 12, 2020	
Getting Hardware	October 13, 2020	October 20, 2020	
Testing of Hardware	October 21, 2020	October 26, 2020	
Circuit Design	October 27, 2020	November 01, 2020	
Interface LED and IR Sensor on BB-WI	November 02, 2020	November 08, 2020	
Interface GSM Module with BB-WI	November 09, 2020	November 17, 2020	
Interface LDR with BB-WI	November 18, 2020	November 22, 2020	
Interface ThingSpeak with BB-WI	November 23, 2020	December 01, 2020	
PCB Designing	December 02, 2020	December 09, 2020	
Final Report Presentation	December 10, 2020	December 14, 2020	
Final Presentation	December 15, 2020	December 17, 2020	

✔ Communication and Programming Standards:

For this project we use the standard communication protocols like the UART (Universal Asynchronous Receiver/Transmitter) . It is a serial communication protocol widely used for many of the microcontrollers. Its main purpose is to transmit and receive the data. This protocol has been used since the olden era through other protocols that still exist like I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface). First let us look at the working of this protocol.

In this protocol the transmitting UART converts the parallel data from a controlling device like a CPU into the serial form to the receiving UART which then converts the serial data into the parallel form. Being asynchronous in nature this protocol doesn't require any clock for synchronizing the output. Thus to synchronise the data it uses the start and the stop bits in the data packet transmission. A point to note is that for a successful communication both the devices must be using the same baud rate with an allowable difference about 10%. It provides a maximum baud rate of 115200 bps but is mostly used at 9600 baud.

Thus we choose this protocol to be used with the GSM Module for the reasons:

1. As we have just one slave device to be considered that is the GSM Module which will help us alert the user
2. Minimum Wiring requirement just two one for the receiving and other for transmitting
3. Also the module that we intend to use has UART pins and no pins for I2C or SPI modes (See the figure 5 for details)
4. Widely used and well documented method when considering GSM for a project.

The primary reason stands that since our peripheral doesn't support any other communication protocol directly thus, we have to use the UART protocol.

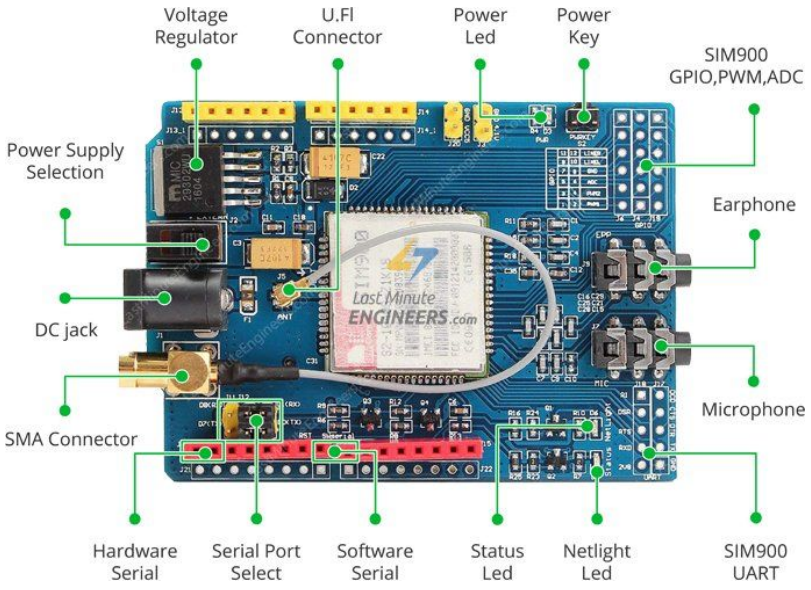


Figure 5: Description for SIM900 GSM Module

Another protocol that we use for this project is the WiFi IEEE 802.11 to be specific we will use the 802.11 ac from the IEEE 802.11 family as this is provided and in-built in the Beaglebone AI which can be seen in the system manual for the same. IEEE 802.11 ac has a baud speed of 1300 Mbps theoretically but one can on a maximum in real world have a speed of 720 Mbps. This protocol will help us to communicate with the cloud with ease and simplicity.

Inorder to transfer the data to the cloud we use the MQTT (Message Queuing Telemetry Transport). As the cloud platform we choose is the ThingSpeak supports HTTP and MQTT we choose the later due to the fact that we need to transfer data and not document on the first instance. Another fact is that it is a lightweight Messaging Protocol along with most popular protocols concerning IoT. It helps devices to send or publish information to a MQTT server that functions as the MQTT message server. It is also a well suited protocol for M2M (Machine to Machine) communication and IoT devices such as real time analysis, monitoring in environments, etc this makes one more reason to use it in our project.

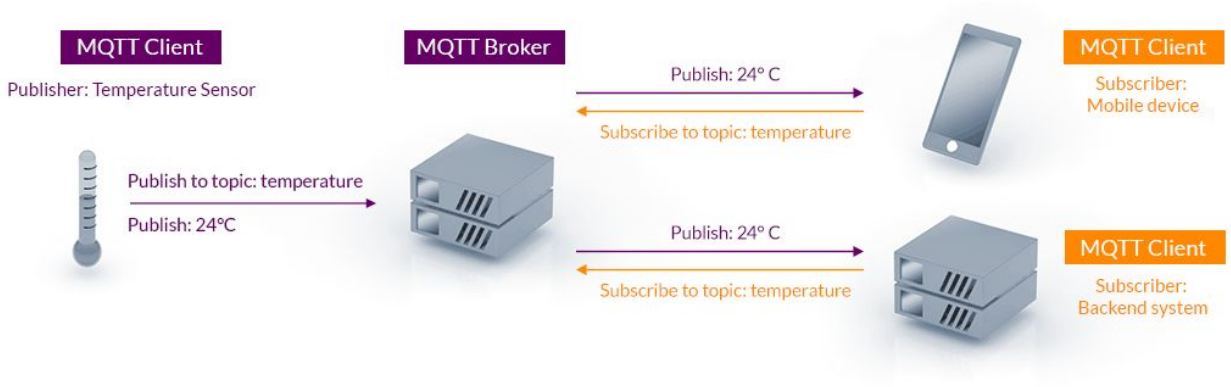


Figure 6: MQTT Functioning

For all other sensors we used the normal GPIOs where we program each of the pins suitably to perform their respective functions. In order to program the pins we need to develop a suitable software code that would be written either in C or C++ with use of python where C or C++ libraries are not present. Thus, we need to follow some coding standards before one can start to write software codes to make things simple and neat to others. We shall now layout some of the standards in coding which we may implement in the project.

1. All programs shall be written to comply with the C99 version of the ISO C Programming Language Standard.
2. All necessary header files must be present before the start of the code with the appropriate descriptions especially if it is a user defined header file.
3. When using parentheses be careful about the operator precedence in C as one cannot rely on the automatic precedence rules
4. If using any threads in the program make sure to mark it with ..._threads (same may be followed for tasks, interrupts or processes). The TAB character shall never appear in the source code files as the width of the tab varies from editor to editor
5. For the codes to be well documented we shall comment the codes as necessary and no code shall be commented out. If required we will use the #if …… #endif. All comments shall be clear and complete sentences with some obvious things exempted. If one needs to highlight a comment then capitalisation may be used

6. Functions having a code length not more than 100 lines with appropriate exit points. Each parameter in the function should be meaningful and explicit.
7. Naming Conventions should be appropriate and the no module name must have any word matching to the preprocessor header and main file names. Underscores may be used in place of space only if naming a file or procedure.
8. No variable name shall be matching any keyword of the C/ Python Language and shall be unique and meaningful with a maximum of 31 characters and if space is required then use underscore for the same. Make sure one initialises this before its use.
9. If one needs infinite loop use for(;;) instead of while(1) to avoid visual confusion; referencing a variable (l). Always use comma only if the variable type is of the same type and behaviour in declarations

In general to speak we shall be using the C18/C11 and C++17 standards to program the GPIO as they are some of the latest standards and have the support of the GCC compiler and most common to use. Also they have a large added features for making coding efficient and optimised; also added are some of the latest in-built functions. If required to use python in the project then python 3 (PEP8) shall be used as we know python is more flexible and is getting updated with libraries and thus we are getting many in-built functions with the standards as mentioned above. The coding must be optimized and reduce the unnecessary occupancy of memory and must be portable as to use on any other platform or OS.

✔ **Ethical, Environmental and Legal Issues Related to the proposed project:**

Every project when created or proposed shall bear some of the challenges and opposition from a group of individuals or the consumers. These may include concerns over the environmental damage being caused, legal issues or some ethical issues. Since our proposed project is like a sub part of a smart home project or a smart city project. Thus, when thinking about the same wouldn't be of that opposition if it stands on its own as a product and sold separately in the market. Some issues or concerns that could be raised include:

1. Since the project uses Internet and sends data to the cloud hence what are the security and privacy features this project shall provide

2. The Data is stored on third party storage like ThingSpeak but when built in the real world may use either company servers or integrate and involve some third party apps etc. So what is the storage security like can the servers be attacked by the hackers and if the data is been collected how will this data be used and will this data be shared with others. This could be a legal and ethical issue.
3. The project involves some sensors like the infrared sensors and the LDR and they may emit some IR rays or some other emission thus people may be concerned how far would it affect the environment and also what would be the power consumption does it contribute to the global warming or will it reduce the global warming effects by features like smart sleep or standby which can help in saving energy.
4. Another issue which may be an ethical issue is that what shall happen when the consumer tries to deactivate or stop the service will the data be retained or shall be deleted completely. If it is retained what is the purpose of it and how shall it be used and what parts of the data be retained.
5. Another ethical issue may arise with respect to the installation of the device for example should the installation be done showcasing where the device is connected and how the device is connected as this may invite danger and also cause to temporary disable the device and then reconnect it after some letters have been stolen from the box or misguide the owner with respect to the count of the letters.
6. Environmental factors could be the use of the material to create the letter box and its design standards. For example would all the boxes be alike and the material used for their construction are safe and would not cause damage to the human and also is the material used is compatible with the electronic used. Can the electronic device set the box on fire? It turns hot like burning the letter in case the device (if hot) and the letter comes in contact.

Since we try to build a simple prototype and this is just like a sub part of a large project to create it wouldn't have many serious concerns or issues to deal with. But yes during the course of the project we can try to make sure that connections are made secure in proper casing and also that the proper cooling is provided in case our main processor gets hot due to the processing of data. Regarding the product security in terms of data since this data is not sensitive thus we can say that we can try to choose a platform which suits the best and gives us better security

authentication. But yes, when this product reaches the market we consider the security aspect more in detail as at this time we are not gathering any information of the letter like from whom and where was the letter sent or received. Thus, much severe concerns may not arise with this project.

✔ **Bill of Materials (BOM) for the proposed Project:**

BILL OF MATERIAL (BOM) FOR THE I-LETTERBOX					
Item #	**Component Name**	**Part Number**	**Qty**	**Description**	**BOM Notes**
1	BeagleBone Wirekess	BBBWL-SC-562-ND *	1	AM5729 BeagleBone AI series ARM® Cortex®-A15 MPU Embedded Evaluation Board.	This a main processing unit which is 64 bit and all the sensors shall be connected to it along with that it would be responsible to send the data to the user via the wifi and GSM Module
2	LED	511-1264-ND *	2	Red 630nm LED Indication - Discrete 2V Radial	This LED will act as an indication for the user to know if the letter is there in the box incase he/she has not used the mobile or logged in the cloud

3	GSM Module	113030009-ND *	1	GPRS SHIELD V3.0, Platform Evaluation Expansion Board	This module will send a SMS/ text message to the user stating that the letter has arrived in the letter box
4	Bread Board	319030002-ND *		BASIC BREAD BOARD 16.5*5.5 CM	Breadboard shall be used to show the prototype before it can be mounted or fixed on the PCB. Like to test if the connection will be proper
5	Light Dependent Resistor (LDR)	PROPR-000100 **	1	This light sensor is based on the GL5528 photoresistor to detect the light intensity of the environment.	The brightness sensor if the letter box is opened in the night time the count of the letter will drop to zero and thus new day/ when the letter is removed the count is initialized again
6	IR (infrared Sensor)	OPTDD-002474 **	1	The GP2Y0A60SZ distance sensor from Sharp offers a wide detection range of 4 to 60 (10 cm to 150 cm) and a high	The IR sensor will help one to count the letter as they keep on arriving in the letter box.

				update rate of 60 Hz. The distance is indicated by an analog voltage, so only a single analog input is required to interface with the module.	
7	Power Adapter	WSU050-1500-13 *	2	5V AC/DC External Wall Mount (Class II) Adapter Fixed Blade Input	Inorder to supply the main processing unit and GSM with the required power we shall use this adapter.
8	Jumper Wires	ED-DP_L20_Mix_120pcs	120	120pcs Breadboard Jumper Wires 10cm 15cm 20cm 30cm 40cm 50cm 100cm Wire Length Optional Dupont Cable Assorted Kit Male to Female Male to Male Female to Female Multicolored Ribbon Cables	To connect the various sensor and other peripherals to the main processing unit we shall use the jumper wires
9	Soldering kit with multimeter	TOL-14681 ***	1	Basic Soldering Equipments Required for any PCB Mounting, etc	Inorder to get the things mounted on the PCB one will use this kit also to check

					if any extra current flows through or if any fluctuation in the currents.

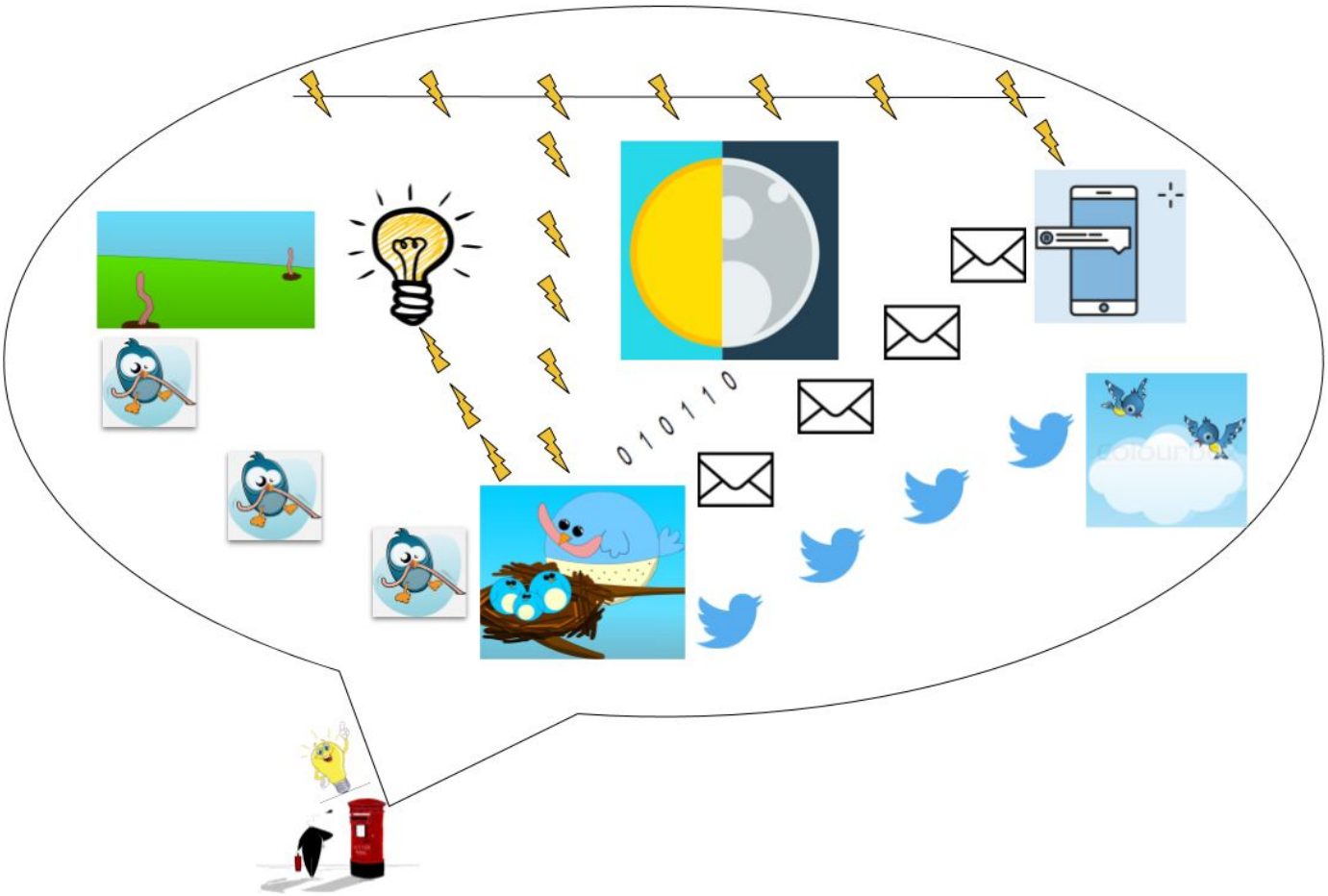
Notes:

** Part Number Adopted from Digikey Website*

*** Part Number Adopted from Creatron Website*

**** Part Number Adopted from Elmwood Electronics Website*

✔ **Engineering Drawing (A prospective from the proposer) :**



✔ **Predicted Edit-Test-Debug Cycle Implementation:**

During any given project one may face some kind of issues and may be required to test and further debug the code to either remove the bugs or get some of the improvements in the code to make better use of memory i.e. optimize the code. So during the execution of this project we may face some of the issues with respect to the testing and debugging of the code. To say we have the five approaches which we already know

1. Using a Target System
2. Using Emulator for Target System
3. Using Target Processor and ICE
4. Using a Simulator for a Hardware
5. Using IDE or Prototyping Tool

We see that we can take any approach that we think is best suited for the project implementation.

Lets see which approaches would best suit our needs when we discuss them one after the other. Talking about the first approach by using a Target System; in our case this target system would be the BeagleBone AI which is built on the ARM Architecture (from the user manual we can see this). Thus we can use this approach and write our code in the host environment i.e. our laptops and using a normal editor like the Nano (incase Linux) or Notepad++ (in case Windows) and compile this code using the cross tool chain compiler like Linaro (needs to be installed and then used) and get the executable file as the output ready after which one just needs this executable file to be present on the target device. To get this file on the target device we have various methods like sending the executable file via the network interface (like using XAMPP or WAMP) or using some of the FTP (File Transfer Protocols) like the SFTP or SCP (Secure Copy Protocol). This approach may be adopted and used during the project. Also we can use the GNU Debugger in case we would like to debug the software.

The next is the emulator for target system approach this may not be feasible enough as the emulator though integrates the hardware and software both in real time but getting the emulator for Beaglebone may be not good as the system that we use is based on Linux Distribution and also the Beaglebone would be of the same environment. Thus we will not adopt any emulator for this project though we could have used Qemu or some other available ones. And since our system shall be compatible with other systems which run C/ C++/ Python codes thus using an emulator doesn't look like a better idea.

Next, one may use the in-built JTAG emulator to debug the codes which may be used in the project but a commitment cannot be made at this point as it requires a use of external software like Code Composer Studio (CCS) which the Beaglebone website also suggests. As in the previous proposal we have already said that there exists no simulation software for the beaglebone device as there is no spice library available for the device and many available simulation software don't support it.

The last approach is using the Integrated Development Environment (IDE) ; this is the most common approach which is used. As IDE usually have their own editor, compiler and debugger. Thus, for the project we may use this approach as well as it suits the best. We mostly use the Eclipse IDE to complete the project/ GNU IDE along with its compiler and debugger inbuilt.

To conclude we may use the approach to get the cross toolchain compiler by building the executable intended to be used only for the ARM-based devices and not any other devices using the Eclipse software and the Linaro Compiler which is the most common industry cross toolchain compiler this may be used incase the where the required libraries cannot be transferred/ not available to Beaglebone. This may be combined and used with the IDE based approach as it will give flexibility for the project and at some point if required we may use the in-built JTAG for the debugging for which the possibility is minimal.

✔ **References:**

Khan, S. (2015, September 23). Intelligent Letter Box using Arduino and GSM. Retrieved September 26, 2020, from https://www.engineersgarage.com/contributions/intelligent-letter-box-using-arduino-and-gsm/

Electronic Letter Box: Embedded Systems Project Topics. (n.d.). Retrieved September 26, 2020, from https://www.seminarsonly.com/Engineering-Projects/Embedded/Electronic-Letter-Box.php

Devi Pujari, A., Bansode, P., Girme, P., Mohite, H., & Pande, A. (2016). Smart Letter Box System Using Obstacle Sensor For Notifies The User By Android Application. *International Research Journal of Engineering and Technology (IRJET), 3*(10), 823-825.

Raithatha, D. (2017, October 02). Smart Letter Box. Retrieved September 26, 2020, from https://www.instructables.com/id/Smart-Letter-Box/

Electronic Letter Box Project Circuit and its Working. (2018, May 26). Retrieved September 26, 2020, from https://www.electronicshub.org/electronic-letter-box-project-circuit/

Light Dependent Resistor Circuit Diagram with Applications. (2019, November 11). Retrieved September 28, 2020, from https://www.elprocus.com/ldr-light-dependent-resistor-circuit-and-working/

BeagleBone AI System Reference Manual. (2020, July 20). Retrieved September 29, 2020, from https://github.com/beagleboard/beaglebone-ai/wiki/System-Reference-Manual

IR (Infrared) Obstacle Detection Sensor Circuit. (2017, December 24). Retrieved September 29, 2020, from https://www.electronicshub.org/ir-sensor/

ThingSpeak. (n.d.). Retrieved September 29, 2020, from https://in.mathworks.com/help/thingspeak/index.html

What is GSM Module and GPRS Module? (2019, January 30). Retrieved September 29, 2020, from https://www.electronicsforu.com/resources/gsm-module

Basics of UART Communication. (2017, April 11). Retrieved October 01, 2020, from https://www.circuitbasics.com/basics-uart-communication/

Yida. (2020, February 03). UART vs I2C vs SPI – Communication Protocols and Uses. Retrieved October 01, 2020, from https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/

Kelly, G. (2019, August 15). 802.11ac vs 802.11n WiFi: What's The Difference? Retrieved October 01, 2020, from https://www.forbes.com/sites/gordonkelly/2014/12/30/802-11ac-vs-802-11n-wifi-whats-the-difference/

Rouse, M. (2020, January 03). What is MQTT and How Does it Work? Retrieved October 01, 2020, from https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport

Beaglebone Forum. (n.d.). Retrieved October 08, 2020, from https://groups.google.com/forum/embed/?place=forum%2Fbeagleboard

Barr, M. (2018). *Embedded C coding standard* (BARR-C:2018 ed.). Estats Units d'Amèrica: Barr Group. Retrieved September 30, 2020, from https://bytebucket.org/HR_ELEKTRO/cxx01/wiki/Documenten/Barr%20Group%202018%20-%20Embedded%20C%20Coding%20Standard.pdf?rev=2532614ccec719079111dcc7bf42eccf90e02ff9

Forbes Technology Council. (2018, January 23). Council Post: 13 Factors To Consider With Smart Home Products. Retrieved October 04, 2020, from https://www.forbes.com/sites/forbestechcouncil/2018/01/23/13-factors-to-consider-with-smart-home-products/

Clever, S., Crago, T., Polka, A., Al-Jaroodi, J., & Mohamed, N. (2018). Ethical Analyses of Smart City Applications. *Urban Science, 2*(4), 96. doi:10.3390/urbansci2040096

Category:Code Composer Studio v5. (n.d.). Retrieved October 08, 2020, from https://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v5

Embedded Software development Embedded Software development Process and Tools:. (n.d.). Retrieved October 08, 2020, from https://d13mk4zmvuctmz.cloudfront.net/assets/main/study-material/notes/computer-science-engineering_engineering_embedded-system-design_embedded-system-development-process-and-tools_notes.pdf

Getting started with JTAG and CCS. (n.d.). Retrieved October 08, 2020, from https://beagleboard.org/static/Docs/ccs-jtag-simple.htm

Instructor's Remarks: