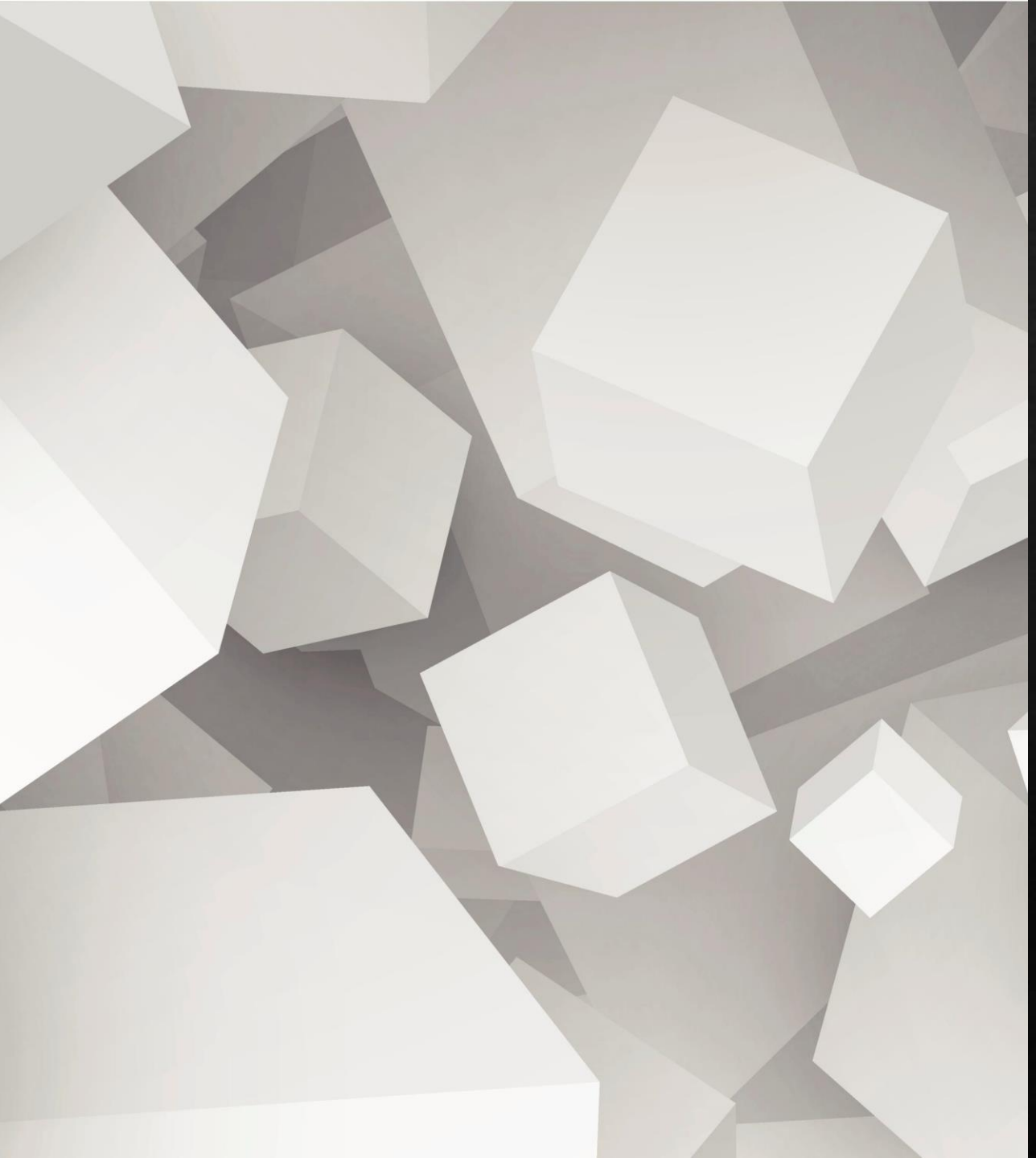
Interfacing with BeagleBone Wireless : Part III

- Zain Rajani (c0752681)

Group # 8



Outline

- Until Now
- Task Objective
- Task Introduction
- Task Requirement
- Interface Programming
- Wire Interface
- Conclusion
- References

Until Now.....

- Until now we have tried to complete the phase I and phase II of the interfacing which was to get the LED, IR sensor merged (interfaced) and GSM with the Beaglebone Black Wireless
- For the interface we used the user made header files and tried to use the POSIX (Portable Operating System Interface) API function library such as the unistd.h (Unix Standard Header) and using the time module for GSM Module
- Using the header files we tried to control the GPIO pins of the BB-WI and made a complete interface in C programming Language.
- For GSM we used python language to send messages to the user and then we integrated this program in our C program using the system(".....") command avaliable in the stdlib.h library of C/C++ language.
- Lastly, we demonstrated the process of interface live and showed that whenever a letter arrives in the letterbox the sensor reads low (being active high in nature) and showed that even if there is a single letter in the letter box the LED will be set high also how the GSM sends message for each letter that arrives.

Task Objective

- To interface the LDR (Light Dependent Resistor) with the Beaglebone Black Wireless (BB-WI)
- To make the coding done in part I and II of the interfacing compatible with the objective mentioned above
- The interfacing should be able to get the counter of the letter to zero when the letter box is opened to remove all the letters*

* *We assume thar the letter box opens from the top and when the letters are deposited it is closed thus the sensor will detect darkness and allow the increment of counter.*

Task Introduction

- In the previous meeting we saw how we broke the interfacing of sensors and peripherials into various tasks.
- In today's task we try to accomplish the part 3 of the interfacing process i.e. we try to interface the LDR Module (**L**ight **D**ependent **R**esistor) with Beaglebone Black Wireless.
- This task is intended to complete in 2 stages first one being just interfacing the BB-WI with LDR Module and in the second stage being trying to accommodate the LDR code into part 1 and part 2 of the interfacing code.
- For this purpose of task we intend to use LDR which has in-built potentiometer to set desired threshold or sensitivity level. The connection is intended to be completed using the GPIO-D (**G**eneral **P**urpose **I**nput **O**utput – **D**igital)
- When the letters are removed from the box that is when the box is opened, we sent the counter back to zero so as when any letter is deposited after that we start from 1

Task Requirements: Hardware

- In order to accomplish this one must have the following hardware present with you:
  - BeagleBone Black Wireless (BB-WI)
  - Connecting Wires
  - Wall Adapters
  - LDR (Light Dependent Resistor) Module
  - Micro USB Cable*
  - Part 1: Sensor and LED**
  - Part 2: GSM Module **

** Required in case of power not sufficient*

*** Required to demonstrate the Part 1, Part 2 and Part 3 together*

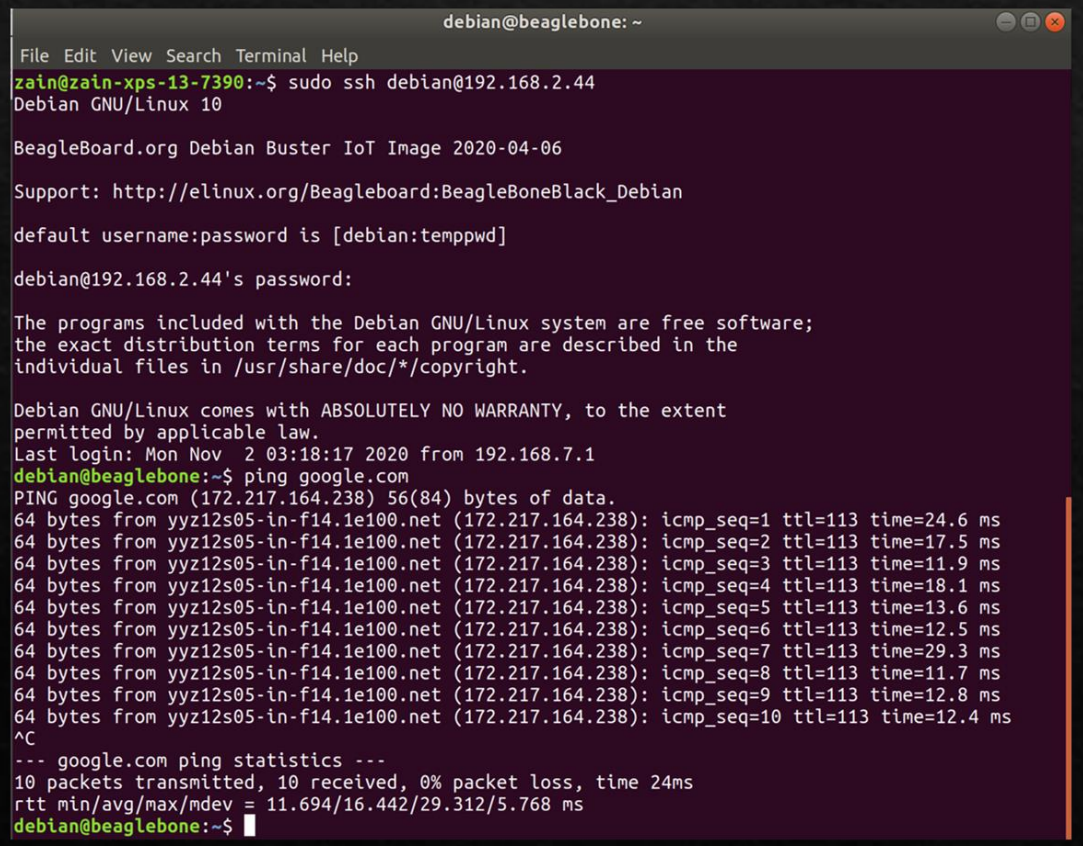
Task Requirements: Software

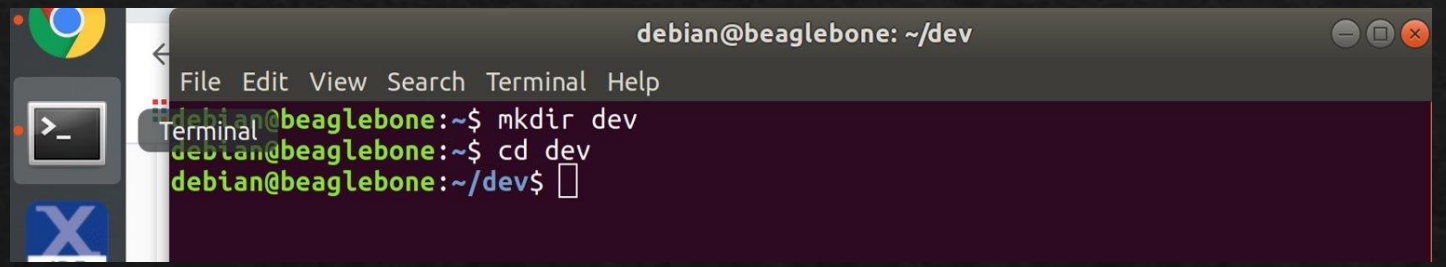
- As for our task objective since we need to program the GSM module to send messages to the desired number via the BeagleBone Black Wireless. For the same reason we need to control the pins thus the pins need to be programmed.
- As mentioned in the proposal and previous meetings we might use Eclipse if the coding cannot be performed on the Beagle bone Wireless itself but since this task can be completed using the nano editor present and the GCC (GNU Compiler Collection) Complier
- To finalize we require the following software tools:
  - Debian OS (Installed on the BB-WI)
  - GCC (GNU [Not Unix] Compiler Collection) to run the code
  - Nano Editor (To write the code)
  - Any required libraries for making interfacing simpler

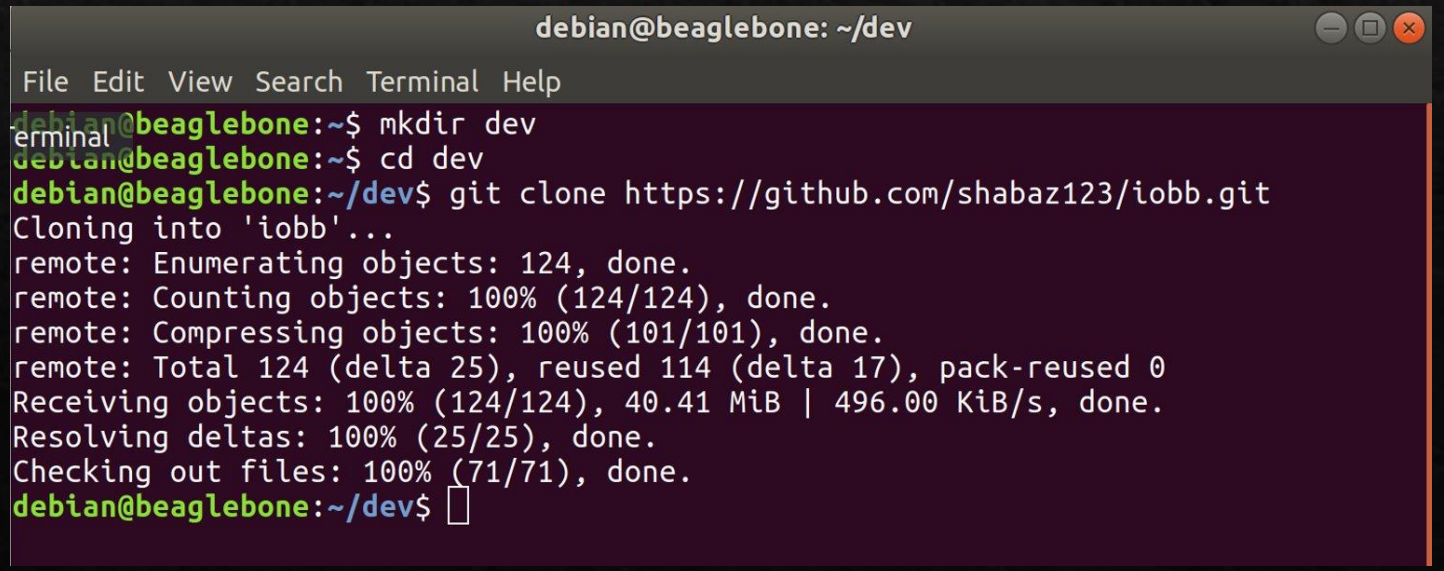
Getting Ready to Code

- Since the user community of the Beaglebone Black Wireless is strong enough thus we have large amount of contribution made each day.
- Thus we have many users which have made our job simpler by giving us some of the functions which help us to gain the control of the GPIOs very easily.
- Only one thing that we require is to install those libraries in the BB-WI and set their paths appropriately. These instructions are usually avaliable when one downloads the files contributed by the user.
- As mentioned in the previous meetings we used some of the user defined header files and their respective functions to gain the control of the GPIO pins and so was the same library files were downloaded and installed on the Beagle bone Black Wireless.
- Thus for this interface we used the same library which we dowloaded for part I while we interaced the IR Sensor and LED with BB-WI

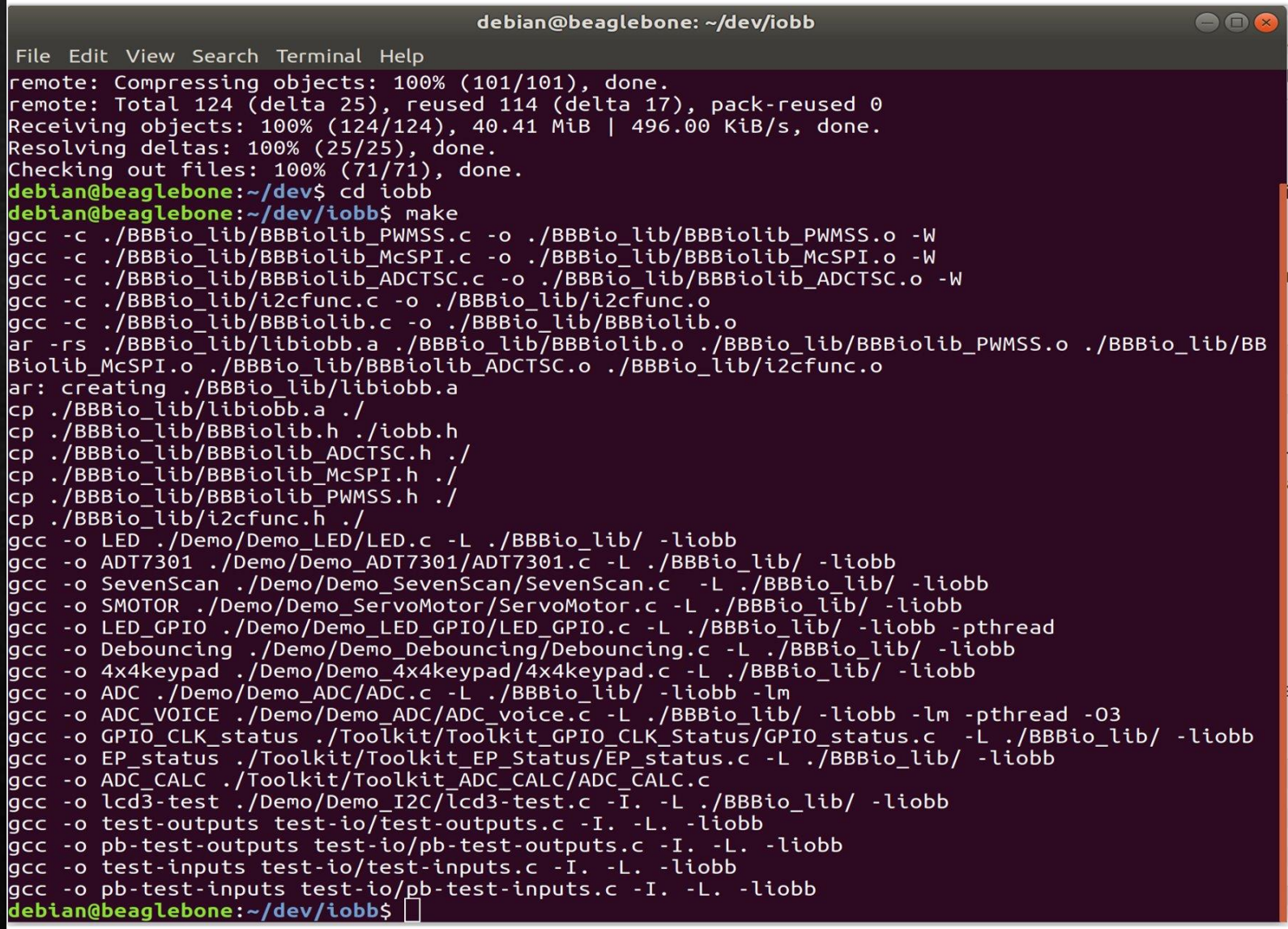
Recap: Library Download



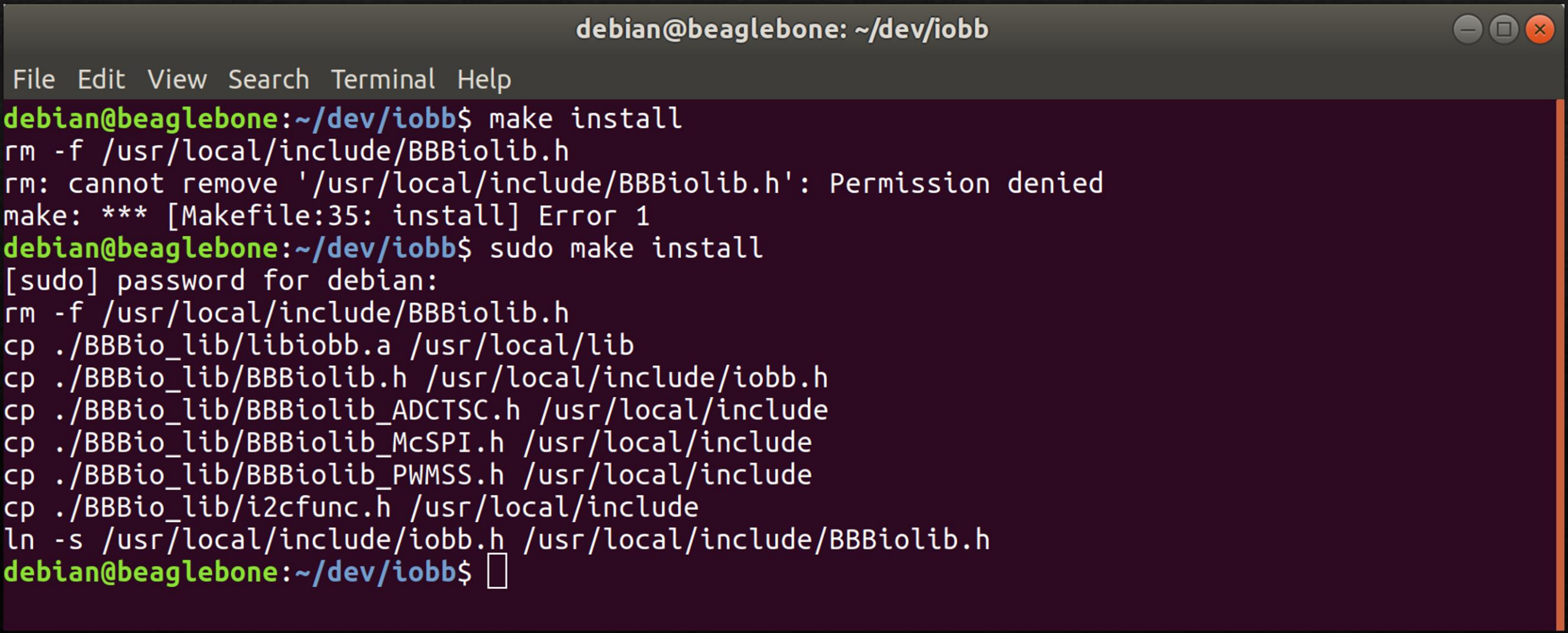




Recap: Library Download



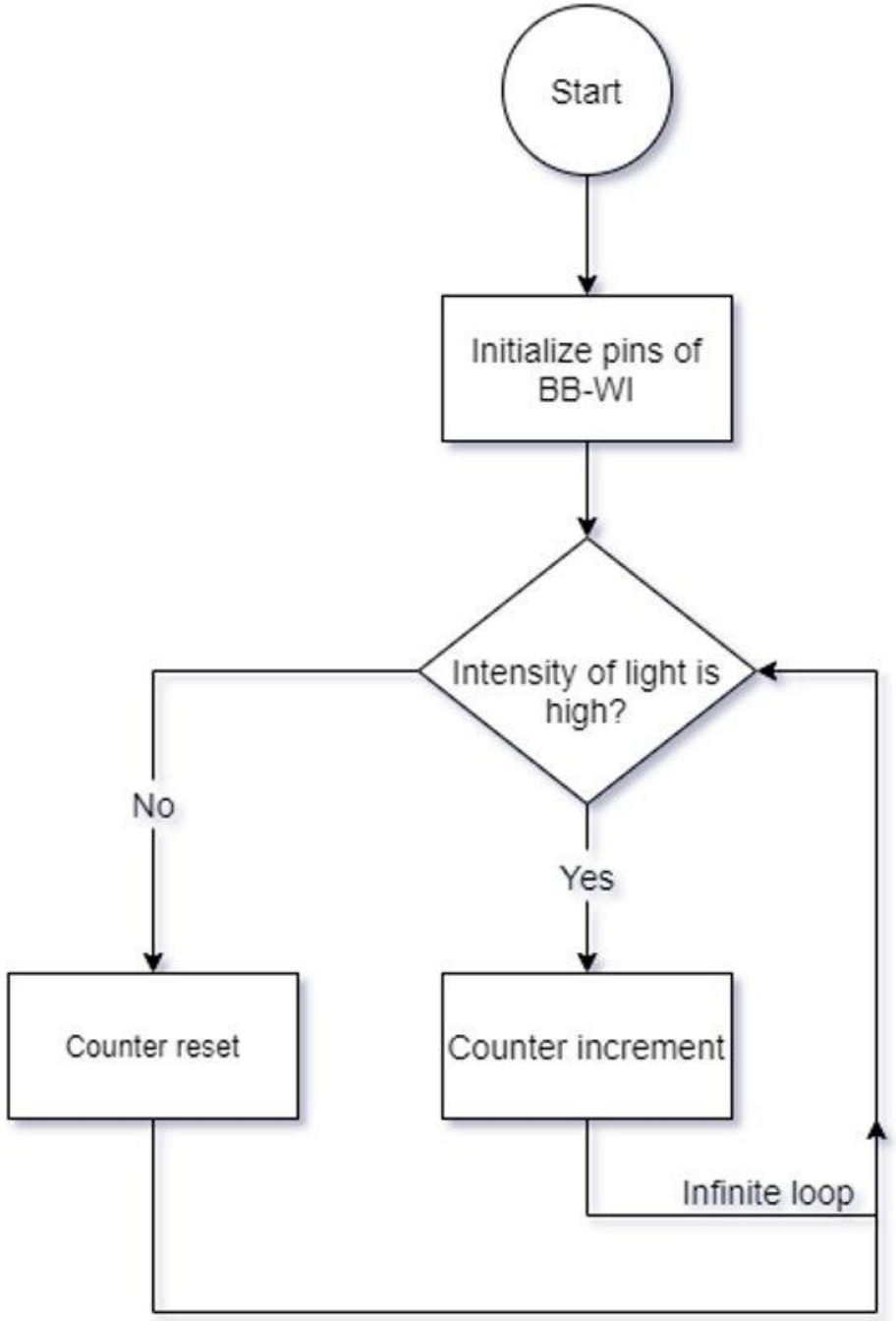
Recap: Library Download



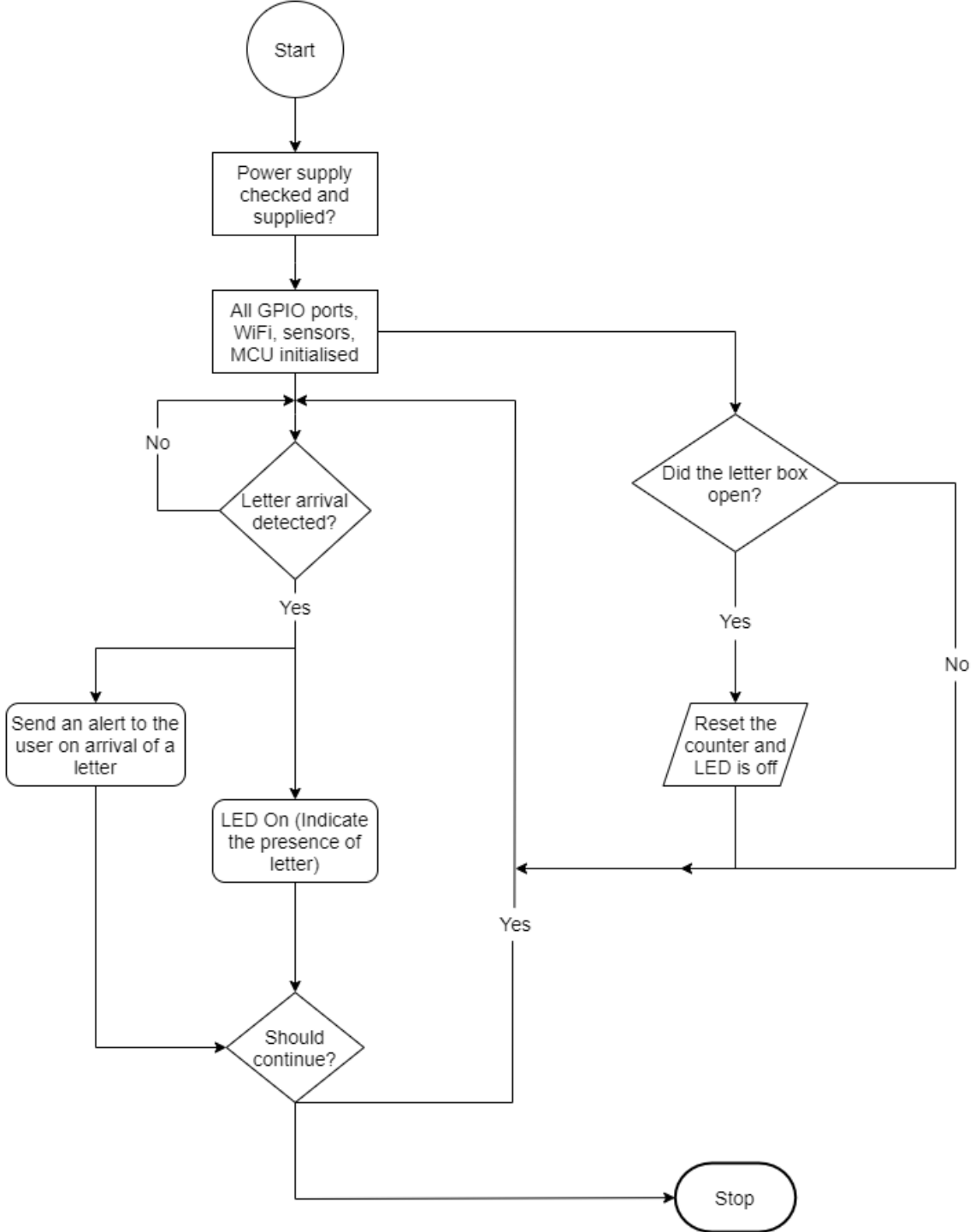
Getting Started: Programming

- A Flowchart describes a process using symbols rather than words. Computer programmers use flowchartsto show and indicate where the data enters, what processes the data will go through and how the data will be converted to the output.
- These flowcharts helps one communicate the ideas or plans that one may visualise, helps in analysis of the process to make sure nothing is left out, provides efficient coding as clarity of data is achieved and it helps programmers figure out potential area where problem may occur in turm helping in debugging (cleaning up code)
- Keeping this ideas in mind we always try to get the logic ready before we actaully code, so we are clear as to what we are planning to do.
- Thus we first plan to get the flowchart for the simple interfacing of LDR with BB-WI and then we accommodate this in the final code for BB-WI

Interface Programming : Flow Chart



Interface Programming : Flow Chart



Interface Programming: Coding

- First step would be to import all the required libraries which will help us gaining the control of the GPIO pins using "**`iobb.h`**", for various input/output function we get the "**`stdio.h`**", to suspend some of the process and conserve some power we may require the usleep(..) or sleep(...) function which is avaliable in the "**`unistd.h`**" and lastly as we must integrate all the code in C thus, we may require to use the system (...) function which is from "**`stdlib.h`**". Thus we import all the four libraries at the beginning of the code.
- The **`unistd.h`** and **`stdlib.h`** both belong to the POSIX API function which is related and used in the real time operations.
- Once all the libraries have been imported we next try to get the code ready as per the flow diagram we designed in the previous steps.

Programming the LDR

debian@beaglebone: ~

File Edit View Search Terminal Help

GNU nano 3.2 LDR_1.c

```
#include <stdio.h>
#include <unistd.h>
#include <iobb.h>

int main()
{
   iolib_init();

   iolib_setdir(9,41,DigitalIn);

   while(1)
   {

   if(is_low(9,41))
   {
      printf("Pin low\n");
      usleep(3000);
   }
   if(is_high(9,41))
   {
      printf("Pin High\n");
      usleep(3000);
   }
   }
   iolib_free();
   return(0);
}
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Programming Part I, II and III of Interfacing

debian@beaglebone: ~

File Edit View Search Terminal Help

GNU nano 3.2 IR_L_GSM.c

```
/* Program to illustrate the interfacing of BB-WI
   and IR (Infrared Sensor) and LED  */

   // Author: Zain Rajani
   // Date: Nov. 01, 2020

/* Libraries to be included here */

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>      // For usleep function
#include <iobb.h>       // To access the GPIO pins easily library already downloaded $

int main()
{
  // Variable Declarations;
  int counter=0;

  //Initialise the GPIO function libraries
  iolib_init();

  //Set the pin function either as input or output pins
  iolib_setdir(9,15,DigitalIn);
  iolib_setdir(8,11,DigitalOut);

  //Setting initially output pin low
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Programming Part I, II and III of Interfacing

debian@beaglebone: ~

File Edit View Search Terminal Help

GNU nano 3.2 IR_L_GSM.c Modified

```
  //Set the pin function either as input or output pins
  iolib_setdir(9,15,DigitalIn);
  iolib_setdir(8,11,DigitalOut);

  //Setting initially output pin low
  pin_low(8,11);          //P8.11 Low (0)

  // Loops forever
  while(1)
  {
    // Check the input value of the pin
    if (is_low(9,15))
    {
      pin_high(8,11);
      system("/usr/bin/python2.7 pysms.py");
      counter++;
      printf("Letter # %d \n",counter);
      usleep(100000);
   }

    if ((is_high(9,15)) && (is_high(9,41)))
    {
      // If sensor detects no object do nothing
      counter=0;
      pin_low(8,11);
    }

  }
// Free up the resources like the GPIO pins for other use if required from memory
iolib_free();

return (0);     // If code success then return 0 else -1
}
```

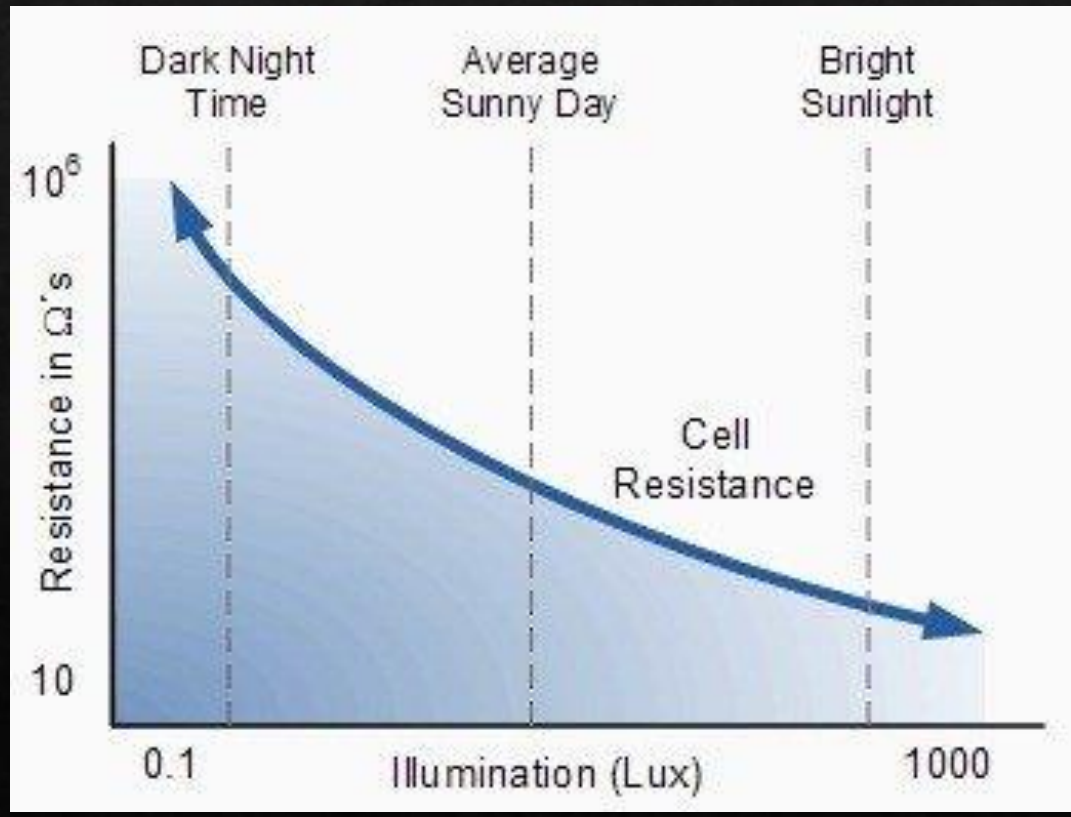
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line

Wire Interface: LDR Module

- Before getting started to wire up the LDR module we must know it well and understand how it works and how it is built.
- The reason one may want to use LDR is stop the wastage of power due to carelessness of human beings or unusual circumstances. We use the advantage of this objective in our project and get the counter to 0.
- To define an LDR or a photoresistor is a device that is made up of high resistance semiconductor material.
- It is basically a photocell that works on the principle of photoconductivity. An LDR is a passive component in nature as it doesn't generate energy.
- An LDR is a type of variable resistor which changes its resistance according to the intensity of light falling on its surface. This sensor senses the intensity of the surrounding light

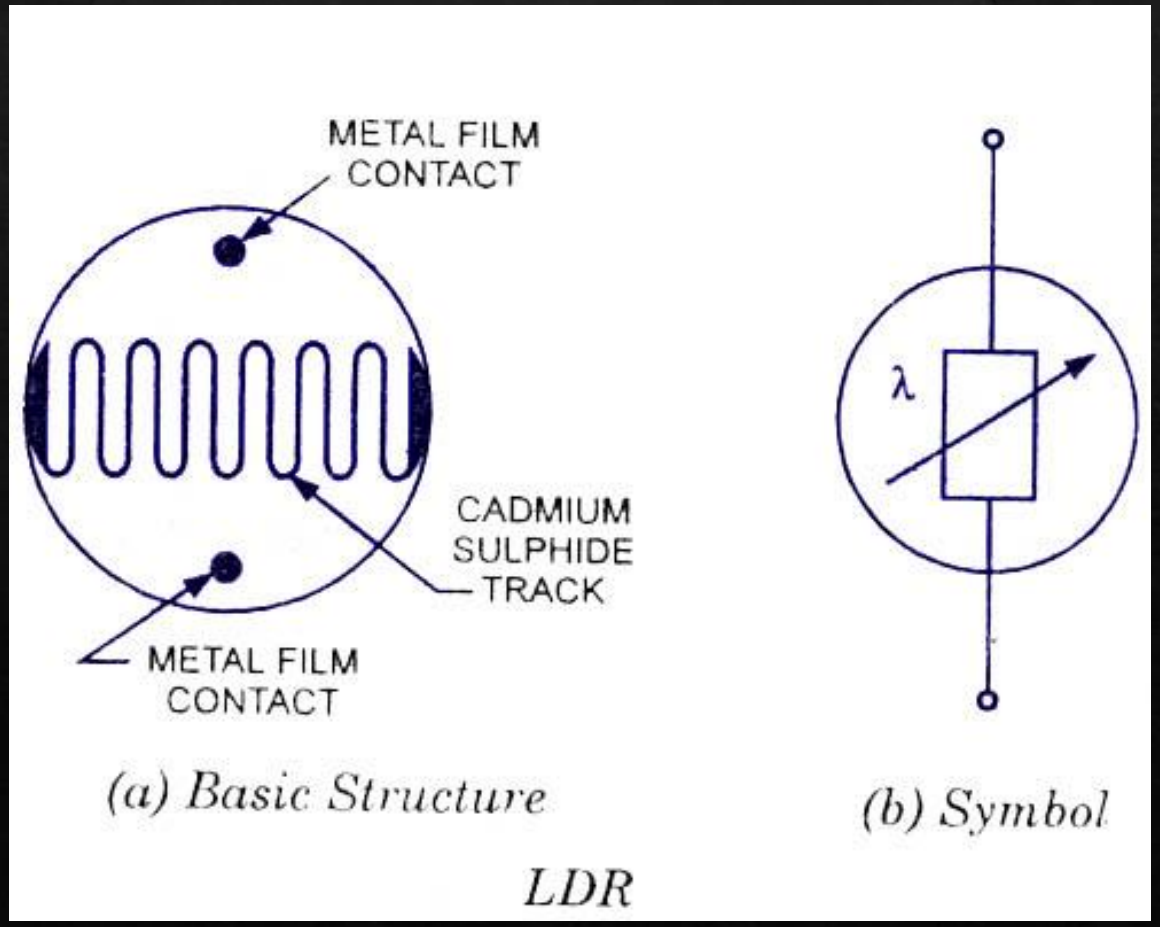
Wire Interface: LDR Module

- Resistance of an LDR is inversely proportional to the intensity of light that falls on LDR's surface. In other words, with an increase in light intensity, the resistance of photoresistor or LDR decreases. And that's why the graph between Resistance of LDR and intensity of light is hyperbolic in nature.
- The construction of an LDR includes a light-sensitive material that is placed on an insulating substrate like as ceramic. The material is placed in a zigzag shape in order to get the required power rating and resistance. The area of zigzag separates the metal placed areas into two regions.

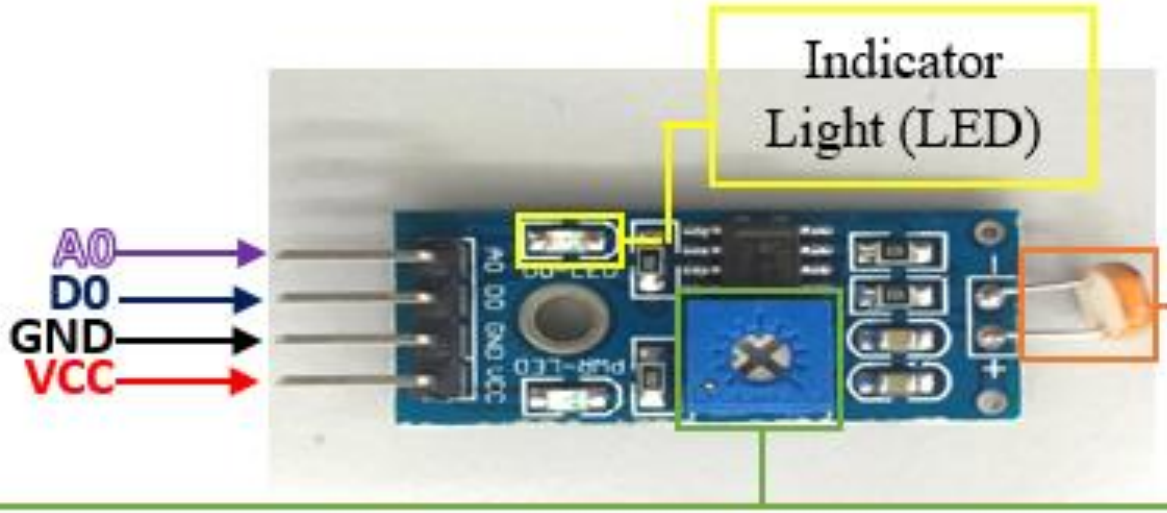


Wire Interface: LDR Module

- The structure is housed in a clear plastic or resin case, to provide free access to external light. As explained above, the main component for the construction of LDR is cadmium sulphide (CdS), which is used as the photoconductor and contains no or very few electrons when not illuminated.
- In the absence of light it is designed to have a high resistance in the range of megaohms. As soon as light falls on the sensor, the electrons are liberated, and the conductivity of the material increases.



Wire Interface: LDR Module



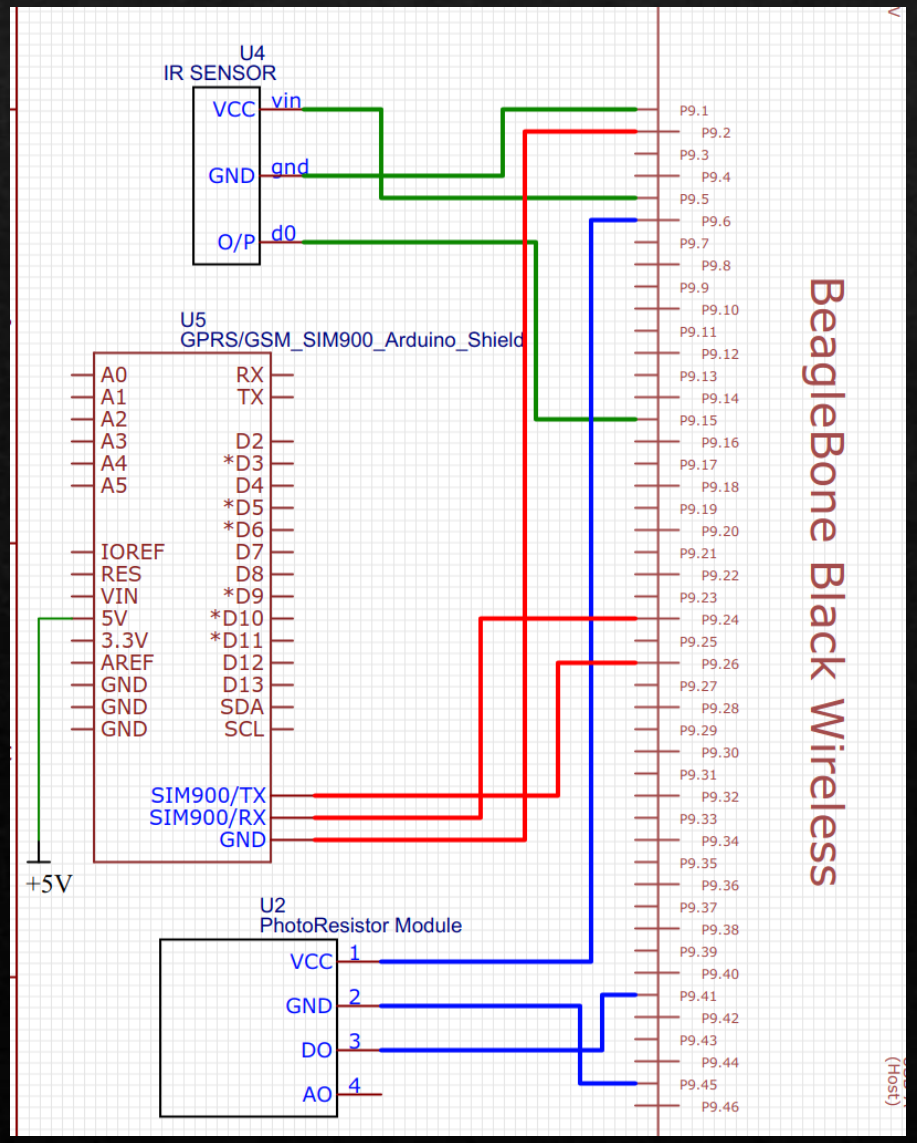
Indicator Light (LED)

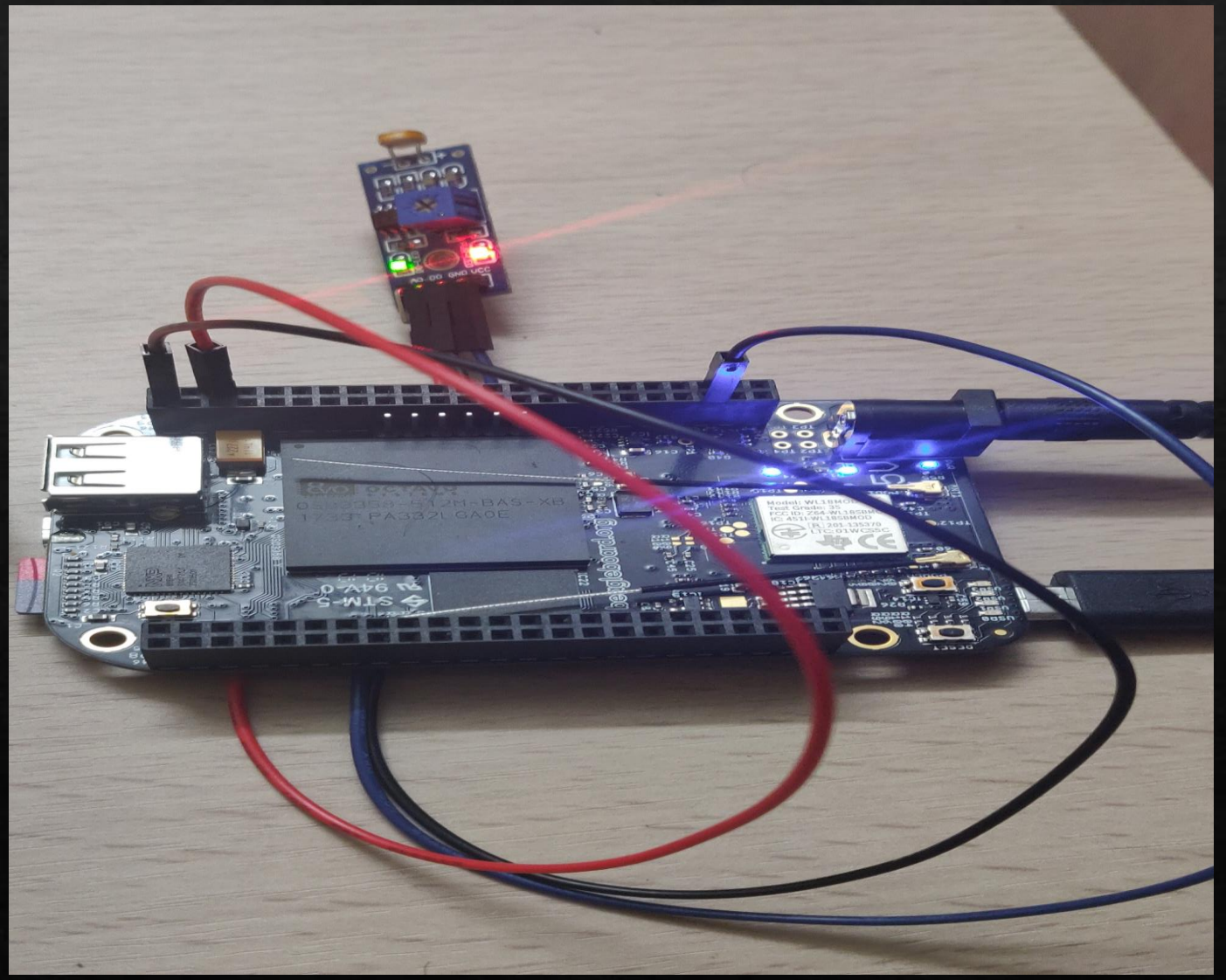
Light Dependant Resistor (LDR)

Sensitivity Adjustor

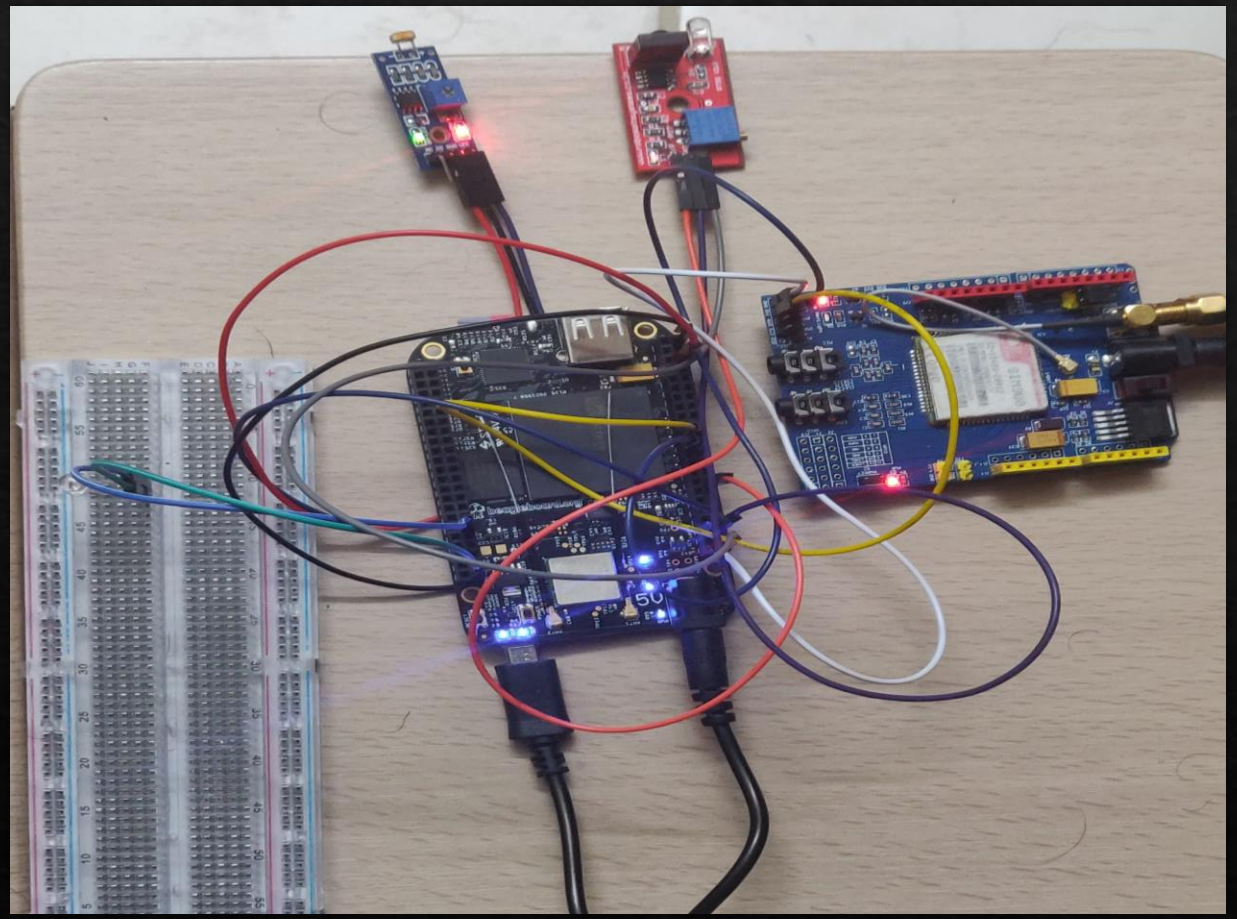
- Clockwise for increasing sensitivity towards intensity of light.
- Anticlockwise for decreasing sensitivity towards intensity of light.

Wire Interface: LDR Module

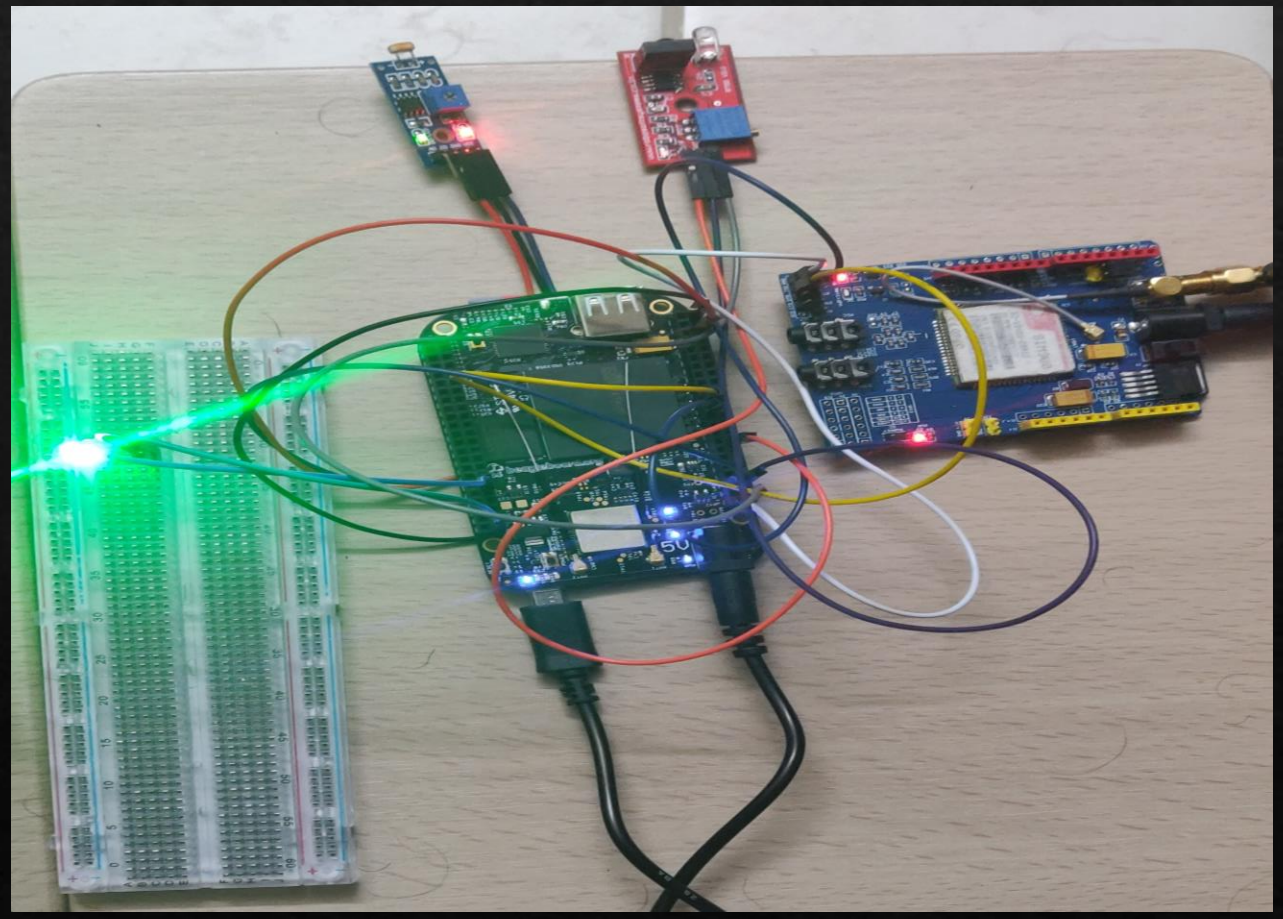




Execution

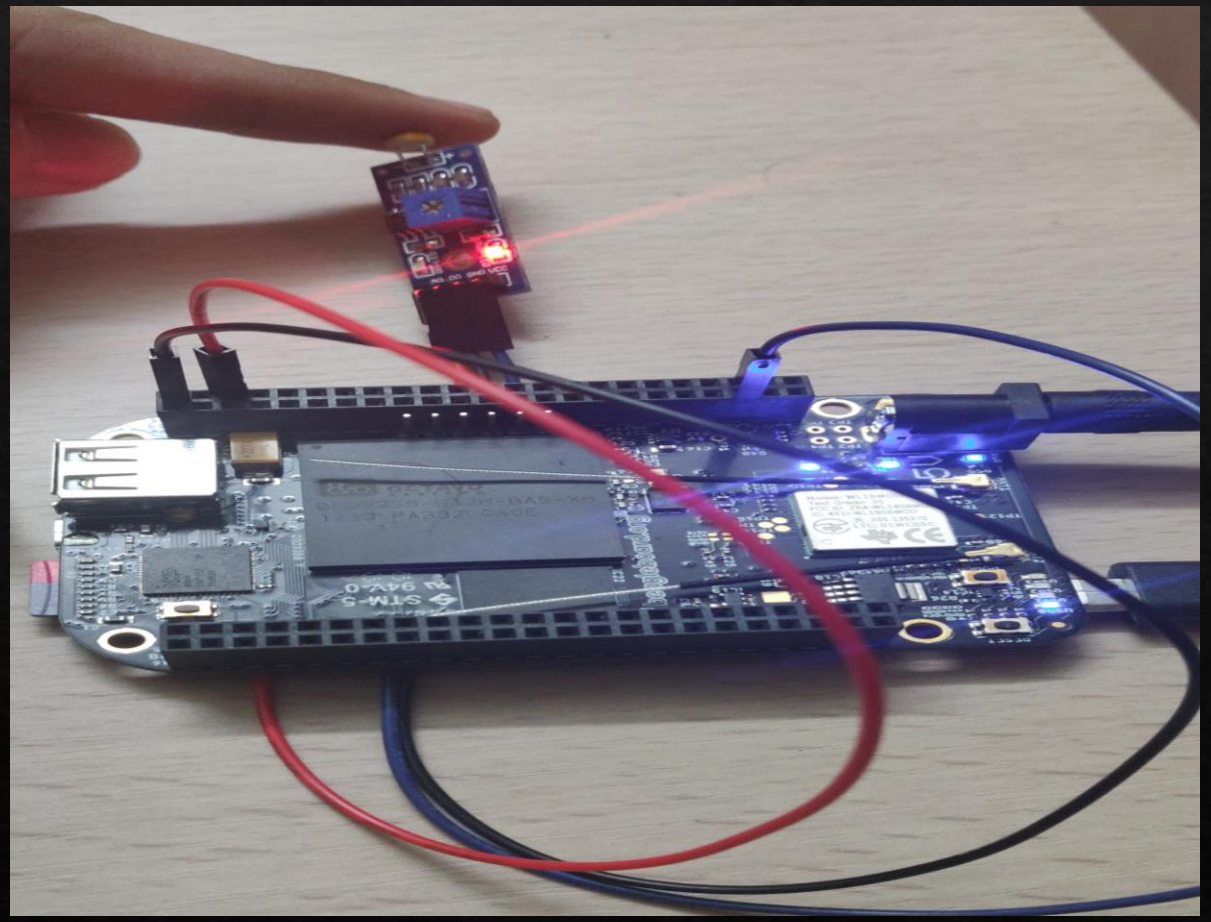


Initial Setup all components ready for its purpose

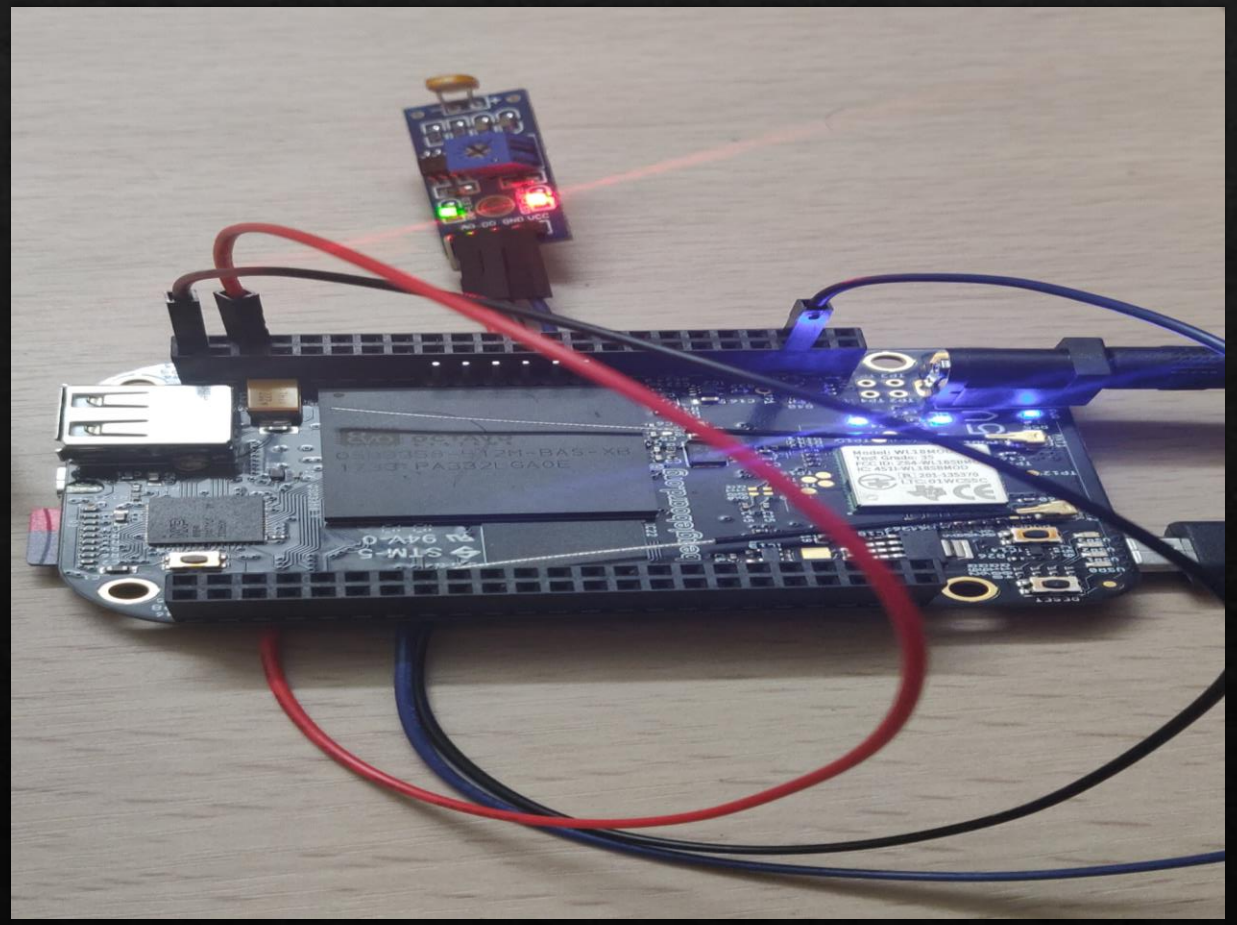


Letter arrived in the box and the brightness is maximum

Execution

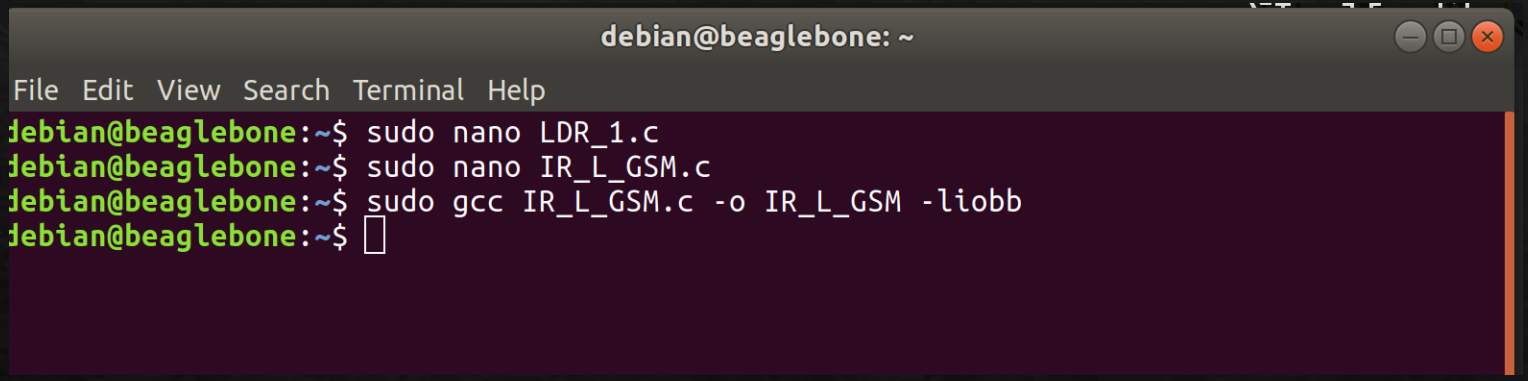


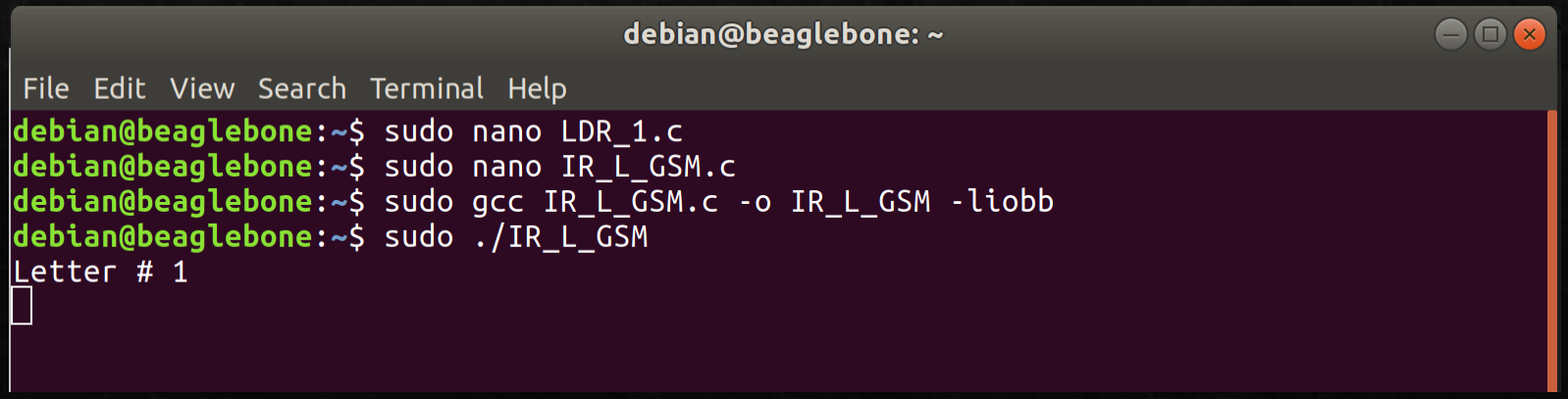
No Brightness on Sensor



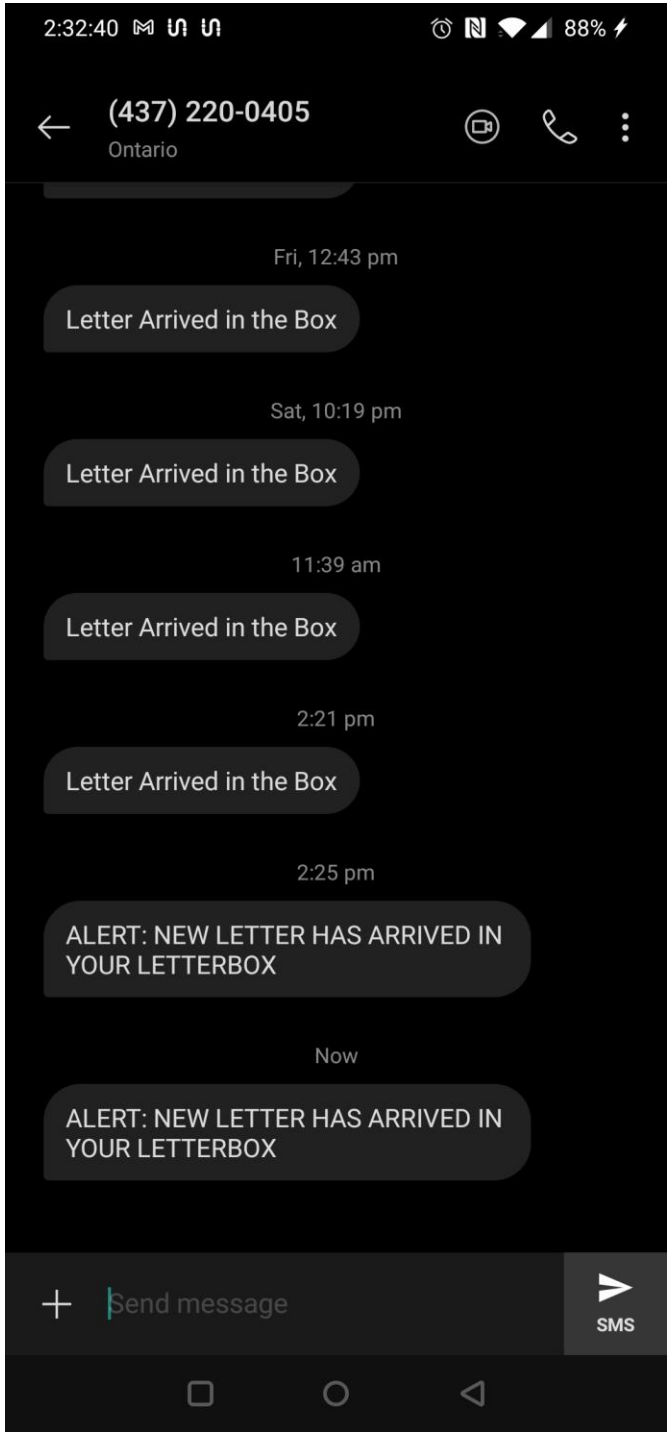
Light Falling on the LDR Material

Execution


debian@beaglebone: ~
File Edit View Search Terminal Help
debian@beaglebone:~$ sudo nano LDR_1.c
debian@beaglebone:~$ sudo nano IR_L_GSM.c
debian@beaglebone:~$ sudo gcc IR_L_GSM.c -o IR_L_GSM -liobb
debian@beaglebone:~$


debian@beaglebone: ~
File Edit View Search Terminal Help
debian@beaglebone:~$ sudo nano LDR_1.c
debian@beaglebone:~$ sudo nano IR_L_GSM.c
debian@beaglebone:~$ sudo gcc IR_L_GSM.c -o IR_L_GSM -liobb
debian@beaglebone:~$ sudo ./IR_L_GSM
Letter # 1

Execution


2:32:40
88%
(437) 220-0405
Ontario
Fri, 12:43 pm
Letter Arrived in the Box
Sat, 10:19 pm
Letter Arrived in the Box
11:39 am
Letter Arrived in the Box
2:21 pm
Letter Arrived in the Box
2:25 pm
ALERT: NEW LETTER HAS ARRIVED IN YOUR LETTERBOX
Now
ALERT: NEW LETTER HAS ARRIVED IN YOUR LETTERBOX
Send message
SMS

Troubleshooting

- This interfacing was simplest of all the interfacing that was performed but there were some issues that we faced which may be considered common issues.
- While programming the interface the sensitivity of the sensor was not set properly thus every time when the light intensity was high or low the sensor displayed as low indicating that the sensor had light falling on it even though at some moment, we didn't have any light falling on the LDR Module.
- To resolve this issue we had turn the potentiometer attached on the module in clockwise direction to make it work properly. After various step of increment we got it functioning as per the need.

Troubleshooting

- While the entire interfacing for this meeting was concentrated on the LDR but also, we had to make it interface with previous meetings thus we had some issues here.
- During the execution of the code we had the IR sensor loop running twice thus when a single letter was given in the box, we had a count of 2 not 1 which was not good for the project. Thus we decided to re-wire complete project and this was completely resolved. So, we assume that we had some wiring issues here.
- Secondly, for the GSM Module since we always must initialize the pins of the UART thus to make our task simple easy and relax free we took the help of the uEnv.txt file where we inserted the line "**`capemgr.enable_partno=BB-UART4`**" and reboot the BB-WI

Conclusion

- This was the last hardware interface that we had to perform with the BB-WI and we have successfully completed this interface
- The module detects the light intensity falling on it in terms of the digital values (0 or 1) which makes our task simpler in resetting the counter
- When interfaced with other hardware components the system works fine and does the following
  - Counts the number of letters arriving (Drawback: No support if 2 letters entered together)
  - Sends SMS to the user indicating the arrival of each letter
  - Gets the counter of the letter to 0 when all the letters are removed from the letterbox

Conclusion

- The system though possess some drawbacks when look at it carefully. One of the drawbacks regarding the entry of the two letters is mentioned.
- Also during the previous meetings we said that the system would be compatible only with certain designs of the letter box. For example for our project we consider that the letter box can only be opened from the top and the letter can arrive in the letter box through a small slit present in the front of the box.
- Thus we have no room to detect any wrong entry of the letter or no space to keep boxes
- Also the system doesn't distinguish between important letters and promotional letters this feature would help the user to respond to important mails earlier if required and he can get them out immediately else mixing up with promotional mails we tend to ignore them.
- Lastly, we have no feature for keeping a record of who deposited the letter in box. This feature may be used in case one receives a threat letter or something which requires investigation or even if one wants to monitor the security of the letter box

References

(n.d.). Retrieved November 16, 2020, from https://study.com/academy/lesson/programming-flowcharts-types-advantages-examples.html

*, N. (2019, November 11). Light Dependent Resistor Circuit Diagram with Applications. Retrieved November 16, 2020, from https://www.elprocus.com/ldr-light-dependent-resistor-circuit-and-working/

Cocker, B. (2018, October 31). Light Dependent Resistors (LDR) - Working,Construction,Symbol,Applications. Retrieved November 16, 2020, from https://www.circuitstoday.com/ldr-light-dependent-resistors

LDR Sensor Module Interface With Arduino. (2017, September 27). Retrieved November 16, 2020, from https://www.instructables.com/LDR-Sensor-Module-Users-Manual-V10/

Molloy, D. (2019). *Exploring BeagleBone: Tools and techniques for building with embedded Linux*. Indiapolis, Indiana: Wiley.

Serial ports / UART. (n.d.). Retrieved November 16, 2020, from http://beaglebone.cameon.net/home/serial-ports-uart

S. (2020, February 13). BeagleBone Black (BBB) and PocketBeagle I/O (GPIO), SPI and I2C Library for C – 2019 Edition. Retrieved November 02, 2020, from https://www.element14.com/community/community/designcenter/single-board-computers/next-genbeaglebone/blog/2019/08/15/beaglebone-black-bbb-io-gpio-spi-and-i2c-library-for-c-2019-edition

Usleep(3) — Linux manual page. (2017, September 15). Retrieved November 01, 2020, from https://man7.org/linux/man-pages/man3/usleep.3.html