Interfacing Proximity IR Sensor with BBB (Detect Garage Status)

Group 4
Dawinder Kaur (c0765505)

Content:

Introduction:

- The previous task was about schematic diagram which include the connection of all components to BBB which is our main processing unit.
- In this task i will explain how i interface the Proximity IR sensor taking reference from the schematic diagram.
- IR sensor plays a very crucial role in our project as it detect the occupancy of garage and detecting the obstacle in between garage door while closing.
- The reason of choosing this interfacing is to detect that garage is occupied or unoccupied.

Required Tools:

To accomplish this task we require following hardware tools:

- Beaglebone Black
- Proximity IR Sensor
- LED
- Jumper cables
- USB cables

Required Tools:

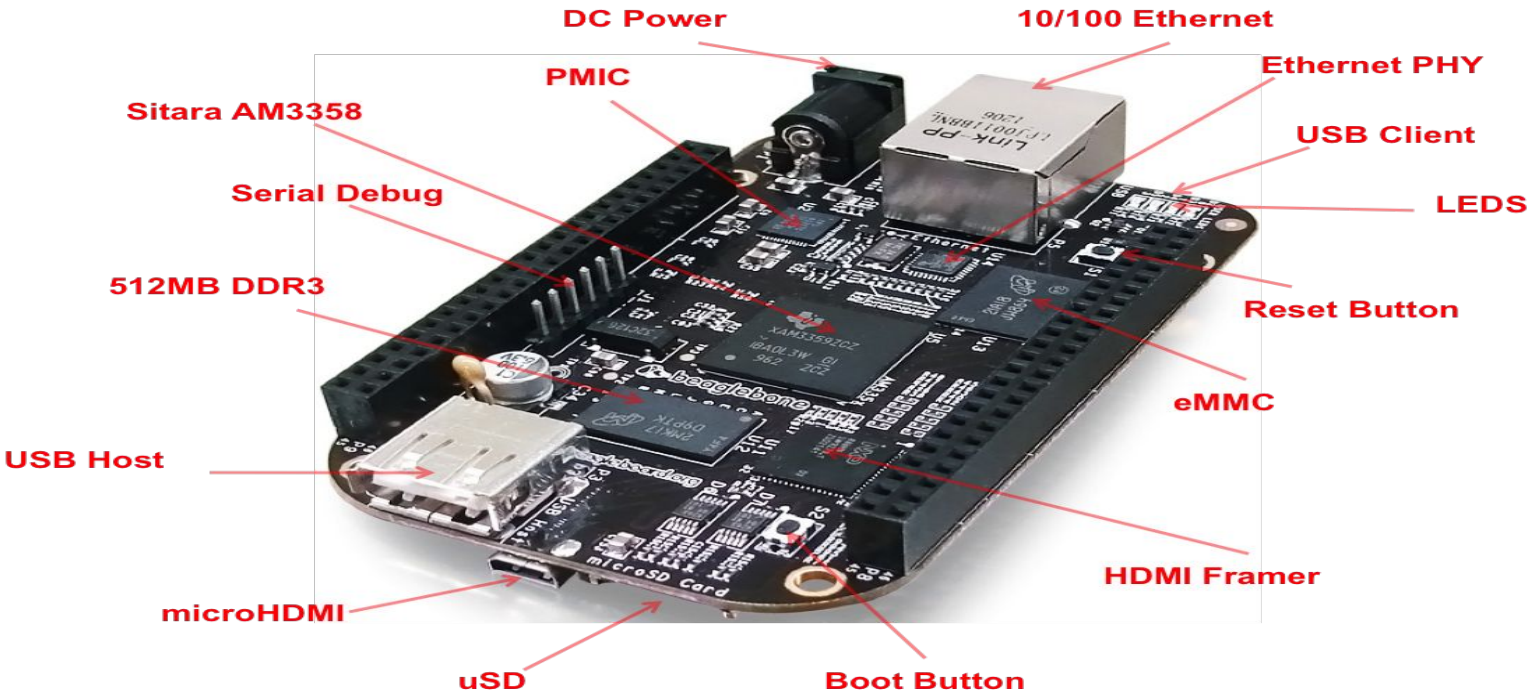
Software requirement:

- Debian OS (Installed on the BBB)
- GCC (GNU Compiler collection) to run the code
- Nano editor (to write the code)
- Any required libraries for making interfacing simpler.

Beaglebone Black:

- BeagleBone Black is a low-cost, community-supported development platform for developers.
- It consist of two header P8 and P9 having numbers of input and output pin.
- These pins are used to interface different type of components.
- It also use different communication protocols to communicate with components like I2C, UART, GPIO etc
- It has AM335x 1GHz ARM Cortex-A8 processor with 512MB DDR3 RAM.
- It also have 4GB 8-bit eMMC on board flash memory.

Beaglebone Black:



Beaglebone Black:

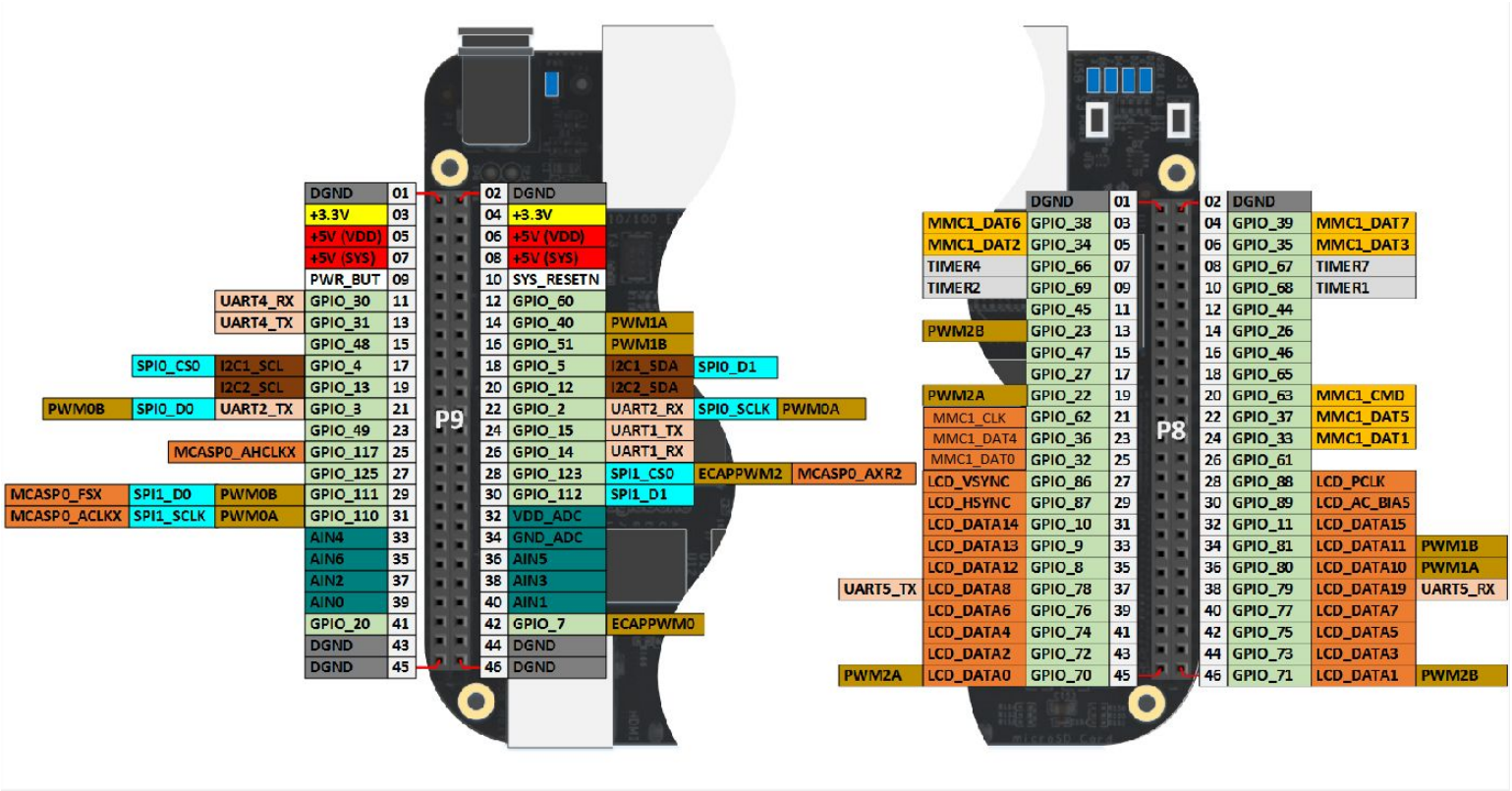
Software compatibility:

- Debian
- Android
- Ubuntu
- Cloud9 IDE on Node.js w/ bonescript library

Connectivity:

- USB client for power & communication
- USB host
- Ethernet
- HDMI
- 2*46 pin header

Beaglebone Black:

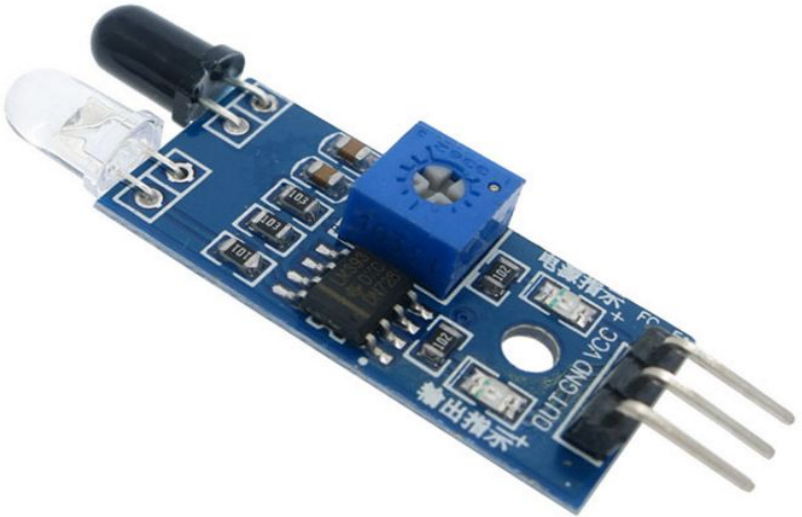


Proximity IR Sensor:

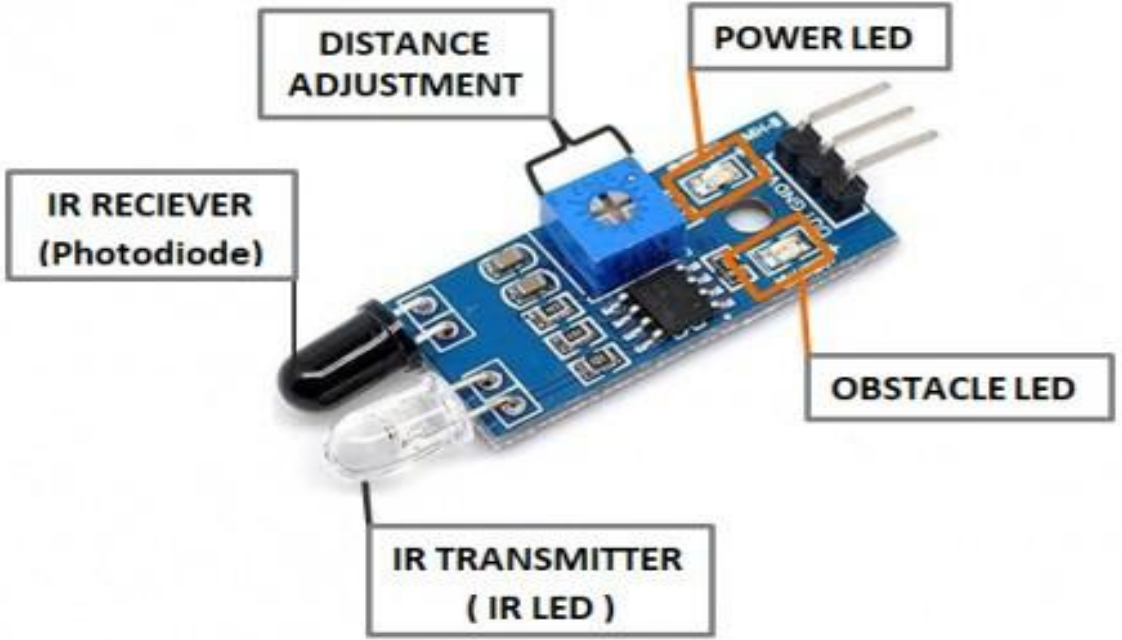
- Proximity IR sensor module has built-in IR transmitter and IR receiver.
- It send out IR energy and looks for reflected IR energy to detect presence of any object.
- This module has onboard potentiometer that lets user adjust detection range.
- This sensor has very good and stable response in ambient light and in complete darkness.

Proximity IR Sensor:

- It has 3 pins as Vcc, GND and an OUTPUT on its tail.
- Working Voltage: DC 3.3V - 5V
- Working Current: >= 20mA
- Detection Distance: 2 - 40 cm
- Output Signal: TTL voltage

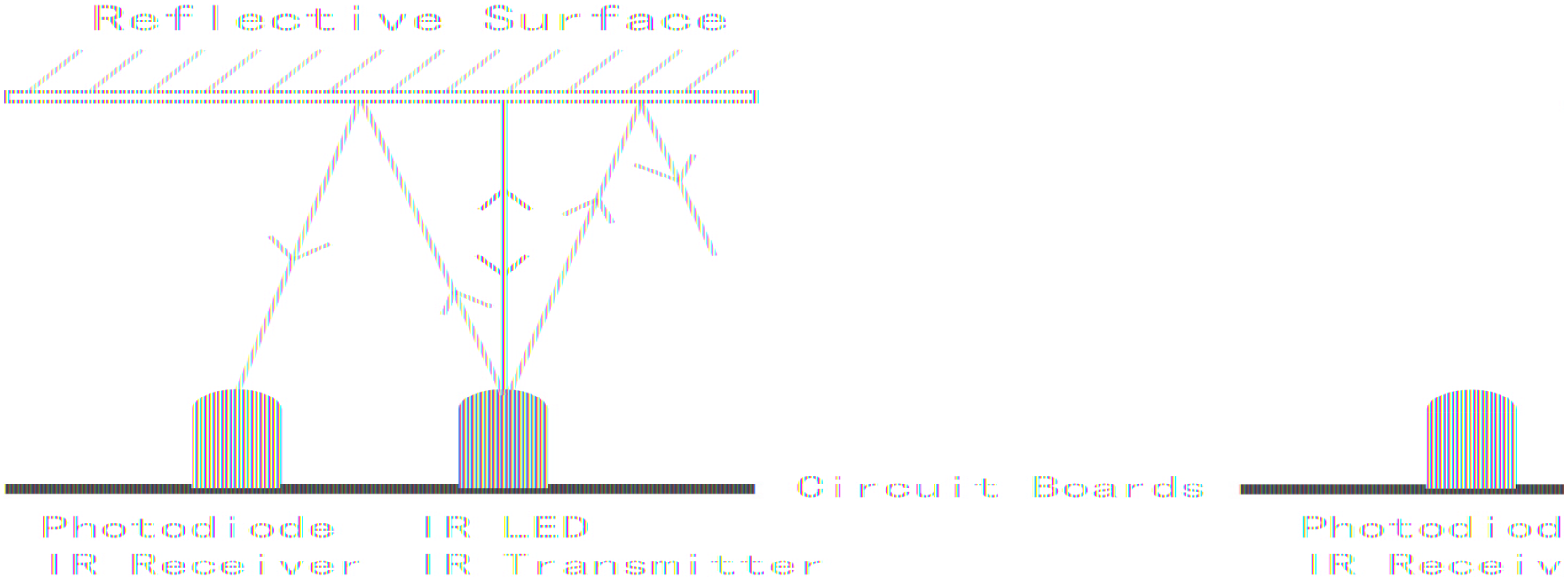


Proximity IR Sensor:



Proximity IR Sensor:

Working or IR Proximity Sensor:



Proximity IR Sensor:

- I use Proximity IR Sensor in two ways in my project

    - Detecting that garage is occupied or vacant
    - Detecting obstacle in-between garage door while closing.

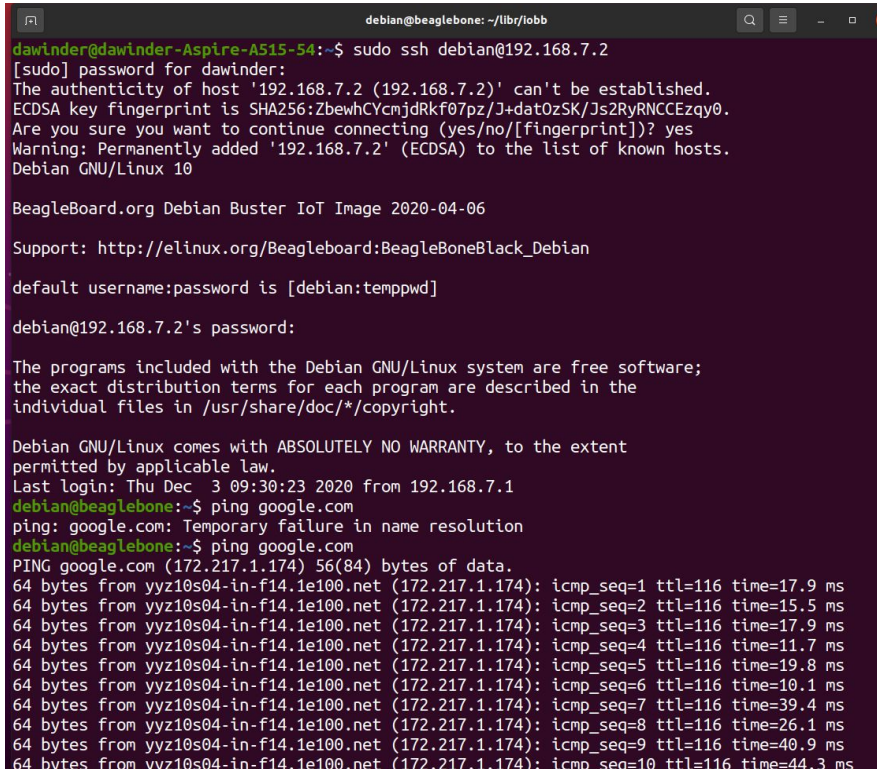
- I will interface proximity IR sensor to detect the garage is occupied or vacant.

Programming:

- Coding Beaglebone Black is very simple as we can write program using different programming languages such as C, C++, Python etc.
- Sometime to write some specific code or to access some specific pins we have to download and install some libraries and headers which are already created.
- Todo the same we also download some user defined header file to access the GPIO pins of the BBB.
- I downloaded the header file name as iobb.h from github and install it in BBB.

Programming: Downloading Libraries:

- Login to BBB using command ssh debian@192.168.7.2
- Check the internet connection
- Ping google.com
- Make directory named libr to add iobb libraries
- Clone git repository for iobb library
- Make and Install the iobb library in BBB



Programming: Downloading Libraries:

```
debian@beaglebone:~$ mkdir libr
debian@beaglebone:~$ cd libr
debian@beaglebone:~/libr$ git clone https://github.com/shabaz123/iobb.git
Cloning into 'iobb'...
remote: Enumerating objects: 124, done.
Receiving objects:  55% (69/124), 38.46 MiB | 2.08 MiB/s  Receiving objects:  56% (70/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  57% (71/124), 39
.58 MiB | 2.09 MiB/s  Receiving objects:  58% (72/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  59% (74/124), 39.58 MiB | 2.09 MiB/s  Receiving obje
cts:  60% (75/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  61% (76/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  62% (77/124), 39.58 MiB | 2.09
 MiB/s  Receiving objects:  63% (79/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  64% (80/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  65% (81/
124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  66% (82/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  67% (84/124), 39.58 MiB | 2.09 MiB/s  Receiv
ing objects:  68% (85/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  69% (86/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  70% (87/124), 39.58 Mi
B | 2.09 MiB/s  Receiving objects:  71% (89/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  72% (90/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:
73% (91/124), 39.58 MiB | 2.09 MiB/s  remote: Total 124 (delta 0), reused 0 (delta 0), pack-reused 124
Receiving objects:  74% (92/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  75% (93/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  76% (95/124), 39
.58 MiB | 2.09 MiB/s  Receiving objects:  77% (96/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  78% (97/124), 39.58 MiB | 2.09 MiB/s  Receiving obje
cts:  79% (98/124), 39.58 MiB | 2.09 MiB/s  Receiving objects:  80% (100/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  81% (101/124), 39.58 MiB | 2.0
9 MiB/s Receiving objects:  82% (102/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  83% (103/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  84% (105
/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  85% (106/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  86% (107/124), 39.58 MiB | 2.09 MiB/s Receiv
ing objects:  87% (108/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  88% (110/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  89% (111/124), 39.58 M
iB | 2.09 MiB/s Receiving objects:  90% (112/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  91% (113/124), 39.58 MiB | 2.09 MiB/s Receiving objects:
92% (115/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  93% (116/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  94% (117/124), 39.58 MiB | 2.09 MiB/
s Receiving objects:  95% (118/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  96% (120/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  97% (121/124),
 39.58 MiB | 2.09 MiB/s Receiving objects:  98% (122/124), 39.58 MiB | 2.09 MiB/s Receiving objects:  99% (123/124), 39.58 MiB | 2.09 MiB/s Receiving ob
jects: 100% (124/124), 39.58 MiB | 2.09 MiB/s Receiving objects: 100% (124/124), 40.41 MiB | 2.13 MiB/s, done.
Resolving deltas: 100% (25/25), done.
Checking out files: 100% (71/71), done.
```

Programming: Downloading Libraries:

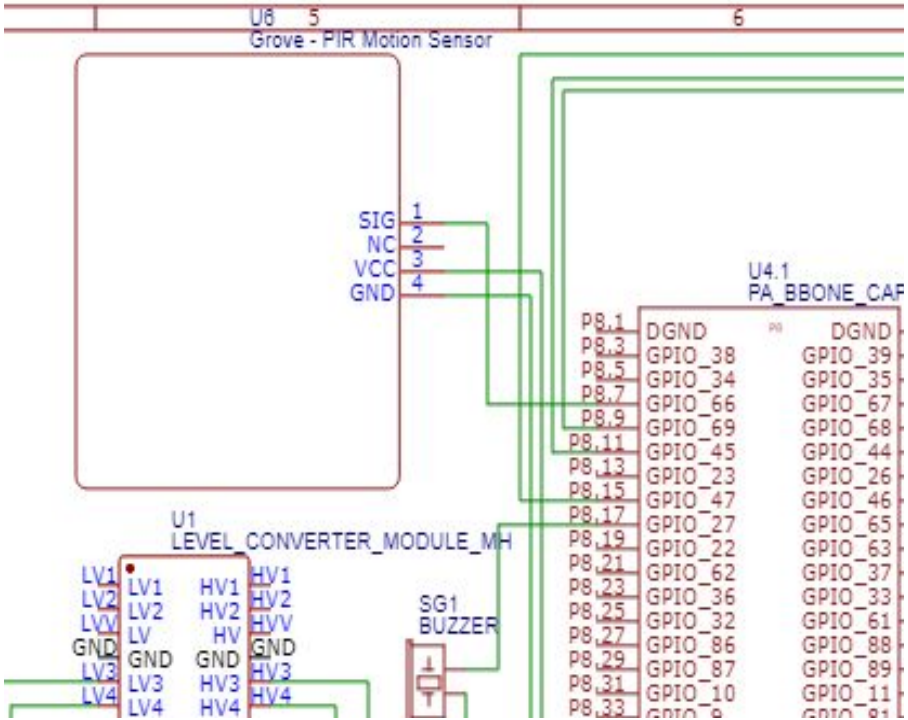
```
debian@beaglebone:~/libr$ cd iobb
debian@beaglebone:~/libr/iobb$ make
gcc -c ./BBBio_lib/BBBiolib_PWMSS.c -o ./BBBio_lib/BBBiolib_PWMSS.o -W
gcc -c ./BBBio_lib/BBBiolib_McSPI.c -o ./BBBio_lib/BBBiolib_McSPI.o -W
gcc -c ./BBBio_lib/BBBiolib_ADCTSC.c -o ./BBBio_lib/BBBiolib_ADCTSC.o -W
gcc -c ./BBBio_lib/i2cfunc.c -o ./BBBio_lib/i2cfunc.o
gcc -c ./BBBio_lib/BBBiolib.c -o ./BBBio_lib/BBBiolib.o
ar -rs ./BBBio_lib/libiobb.a ./BBBio_lib/BBBiolib.o ./BBBio_lib/BBBiolib_PWMSS.o ./BBBio_lib/B
i2cfunc.o
ar: creating ./BBBio_lib/libiobb.a
cp ./BBBio_lib/libiobb.a ./
cp ./BBBio_lib/BBBiolib.h ./iobb.h
cp ./BBBio_lib/BBBiolib_ADCTSC.h ./
cp ./BBBio_lib/BBBiolib_McSPI.h ./
cp ./BBBio_lib/BBBiolib_PWMSS.h ./
cp ./BBBio_lib/i2cfunc.h ./
gcc -o LED ./Demo/Demo_LED/LED.c -L ./BBBio_lib/ -liobb
gcc -o ADT7301 ./Demo/Demo_ADT7301/ADT7301.c -L ./BBBio_lib/ -liobb
gcc -o SevenScan ./Demo/Demo_SevenScan/SevenScan.c  -L ./BBBio_lib/ -liobb
gcc -o SMOTOR ./Demo/Demo_ServoMotor/ServoMotor.c -L ./BBBio_lib/ -liobb
gcc -o LED_GPIO ./Demo/Demo_LED_GPIO/LED_GPIO.c -L ./BBBio_lib/ -liobb -pthread
gcc -o Debouncing ./Demo/Demo_Debouncing/Debouncing.c -L ./BBBio_lib/ -liobb
gcc -o 4x4keypad ./Demo/Demo_4x4keypad/4x4keypad.c -L ./BBBio_lib/ -liobb
gcc -o ADC ./Demo/Demo_ADC/ADC.c -L ./BBBio_lib/ -liobb -lm
gcc -o ADC_VOICE ./Demo/Demo_ADC/ADC_voice.c -L ./BBBio_lib/ -liobb -lm -pthread -O3
gcc -o GPIO_CLK_status ./Toolkit/Toolkit_GPIO_CLK_Status/GPIO_status.c  -L ./BBBio_lib/ -liobb
gcc -o EP_status ./Toolkit/Toolkit_EP_Status/EP_status.c -L ./BBBio_lib/ -liobb
gcc -o ADC_CALC ./Toolkit/Toolkit_ADC_CALC/ADC_CALC.c
gcc -o lcd3-test ./Demo/Demo_I2C/lcd3-test.c -I. -L ./BBBio_lib/ -liobb
gcc -o test-outputs test-io/test-outputs.c -I. -L. -liobb
gcc -o pb-test-outputs test-io/pb-test-outputs.c -I. -L. -liobb
gcc -o test-inputs test-io/test-inputs.c -I. -L. -liobb
gcc -o pb-test-inputs test-io/pb-test-inputs.c -I. -L. -liobb
debian@beaglebone:~/libr/iobb$
```

```
debian@beaglebone:~/libr/iobb$ sudo make install
[sudo] password for debian:
Sorry, try again.
[sudo] password for debian:
rm -f /usr/local/include/BBBiolib.h
cp ./BBBio_lib/libiobb.a /usr/local/lib
cp ./BBBio_lib/BBBiolib.h /usr/local/include/iobb.h
cp ./BBBio_lib/BBBiolib_ADCTSC.h /usr/local/include
cp ./BBBio_lib/BBBiolib_McSPI.h /usr/local/include
cp ./BBBio_lib/BBBiolib_PWMSS.h /usr/local/include
cp ./BBBio_lib/i2cfunc.h /usr/local/include
ln -s /usr/local/include/iobb.h /usr/local/include/BBBiolib.h
debian@beaglebone:~/libr/iobb$
```

Interfacing PIR sensor:

Schematic connection of PIR sensor with BBB:

PIR Sensor	Beaglebone Black
SIG/OUT	GPIO_66 (P8.7)
VCC	3V
GND	GND

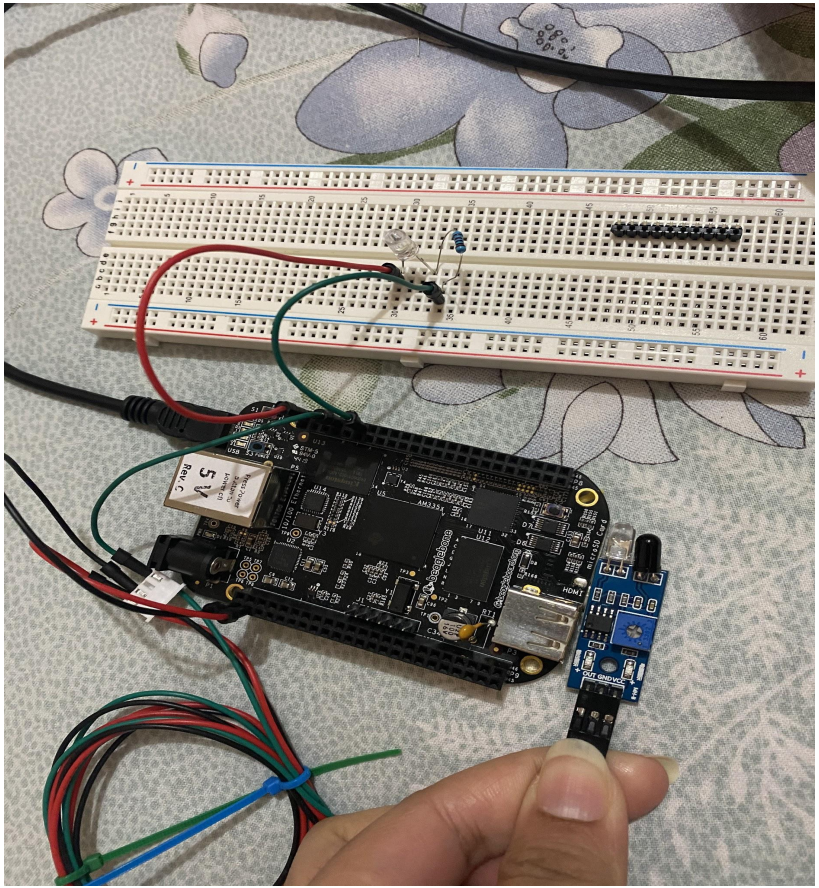


Interfacing PIR Sensor:

Connection of PIR sensor and LED

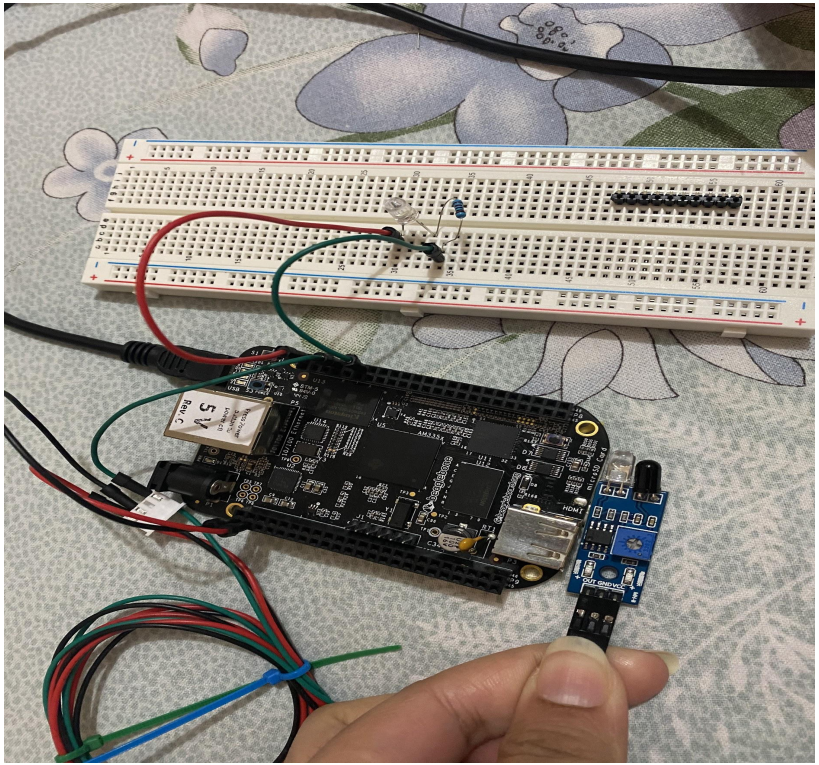
LED is connected only for demonstration.

- LED anode pin (+) to pin 12 of P8 BBB
- 1k ohm resistor connected to anode pin of LED
- LED cathode pin (-) to pin 2 (GND) of P8 BBB

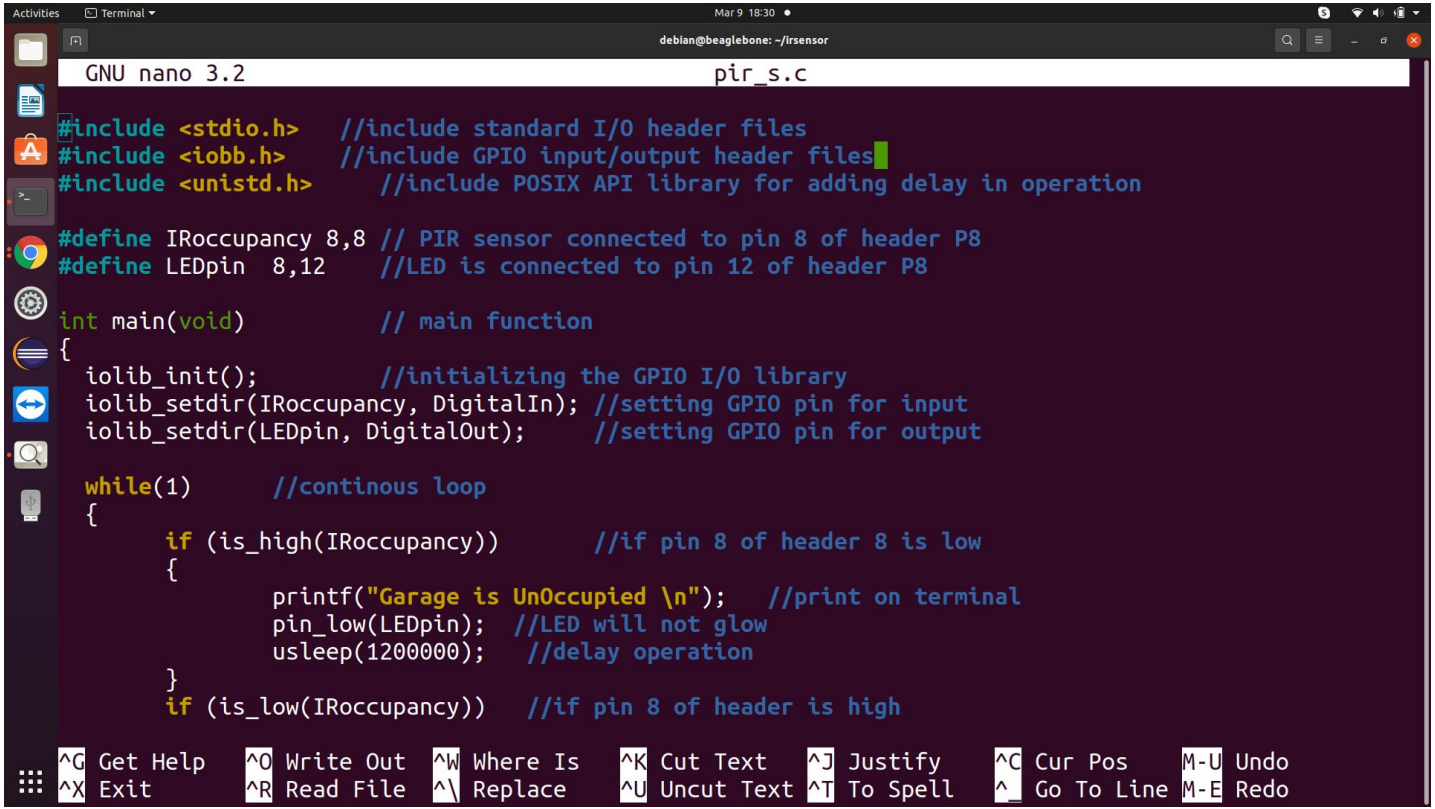


Interfacing PIR Sensor: Coding

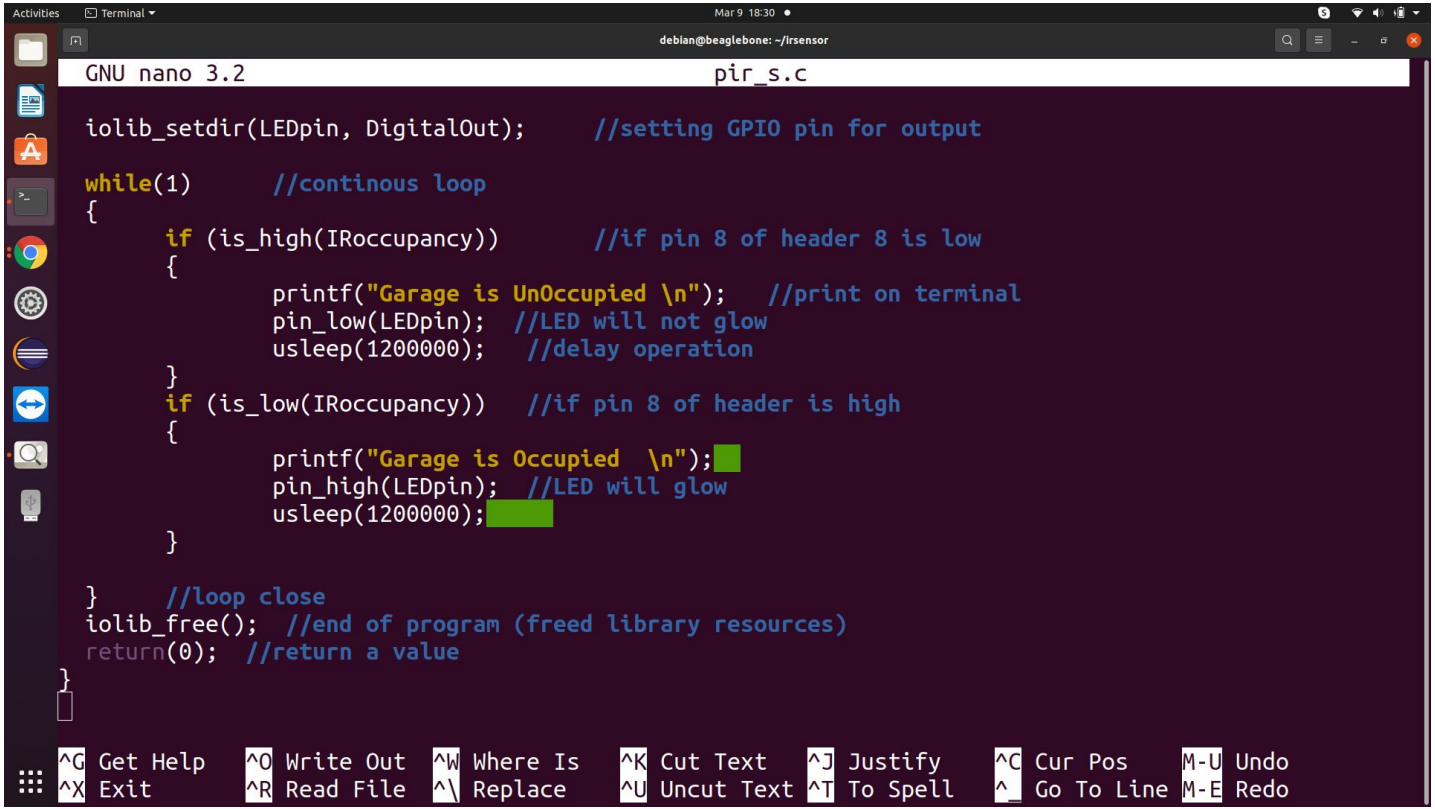
- Connect to BBB using ssh debian@192.168.7.2 command
- Enter the password : temppwd
- Open the nano terminal using nano pir_s.c command
- Write the c code
- Press ctrl+O to save and press enter
- Press ctrl+X to exit the nano terminal



Interfacing PIR Sensor: Coding



Interfacing PIR Sensor: Coding



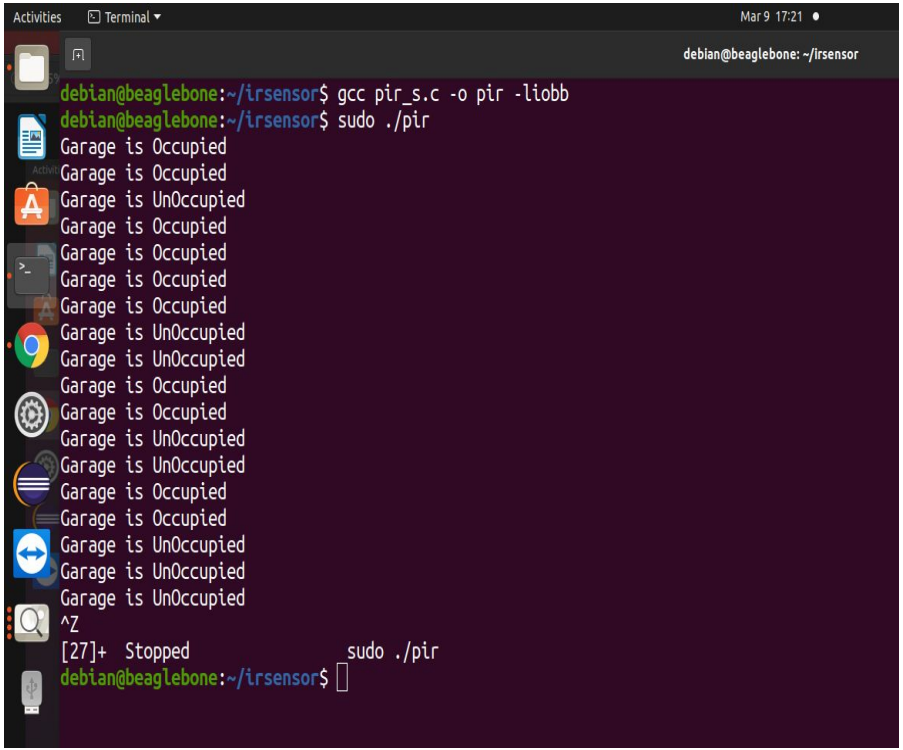
Interfacing PIR Sensor: Execution

For compilation :

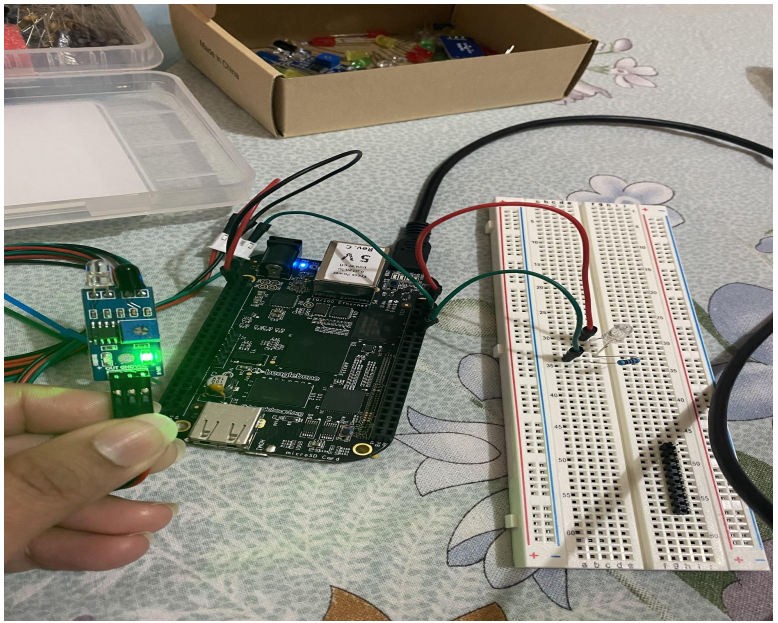
- GCC - GNU compiler collection for C/C++
- -o pir - object file named pir created
- -liobb - adding GPIO I/O library

For execution:

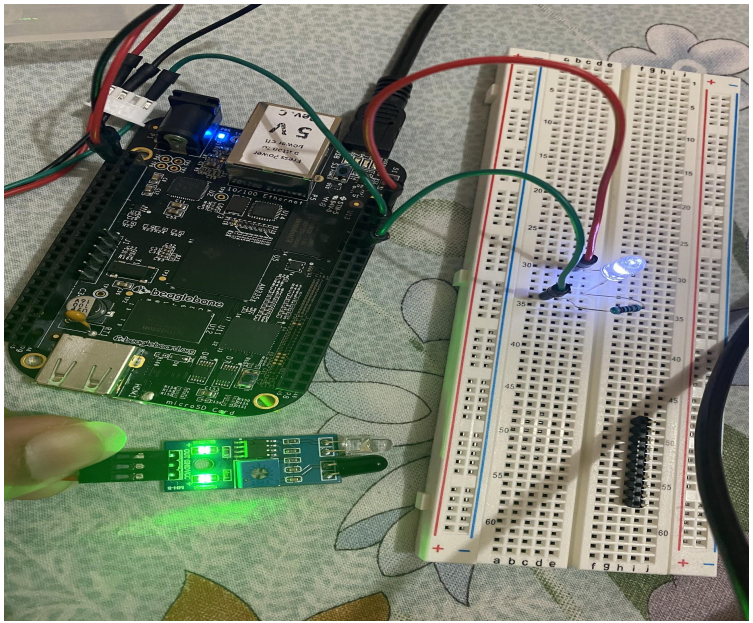
- sudo ./pir - executing object file pir as a root



Interfacing PIR Sensor: Output



When garage is Unoccupied the LED will not Glow



When garage is occupied the LED will Glow

Troubleshooting:

1. While cloning a git repository for GPIO library. I forget to connect the BBB to internet via ethernet cable.
2. While writing c code. The output is shown as continuous flow on the terminal.
3. Use POSIX API library to add delay during suspending operations
4. LED was not glowing at first as i connect it to pin 11 of header P8 but after changing it to pin 12 of header P9 it start glowing.
5. In c programming while defining LED pins i defined it as "*#define LEDpin 12,8"* i got error "*segmentation fault*" . then i changed it to "*#define LEDpin 8,12"*

Conclusion:

Task is performed successfully

Resolve all the error found in c program

PIR sensor is showing accurate status of garage, either it is occupied or unoccupied.

This status will be shown on HTML page.

References:

Beaglebone.org (2019,September 6). Retrieved from https://beagleboard.org/black

Arduino lesson- IR obstacle Avoidance sensor, Osoyoo (2017,july 21). Retrieved from https://osoyoo.com/2017/07/24/arduino-lesson-obstacle-avoidance-sensor/

Shabaz Github (2019, August 17). Retrieved from https://github.com/shabaz123/iobb

Shabaz (2019,August 14) Element14.com Retrieved 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 Linux manual page (2017,September 15) Retrieved from https://man7.org/linux/man-pages/man3/usleep.3.html

References:

Monk S. (2021) Blinking an LED with BeagleBone Retrieved from Blackhttps://learn.adafruit.com/blinking-an-led-with-beaglebone-black/writing-a-program