Practical No :- 1

Aim :-

Study of IoT 3500 development kit and familiarization with Intel Galileo board and perform necessary software installations.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4 GB+RAM, 1 TB HDD	1
3	Bootable Pen drive	32 GB	1

Theory :-

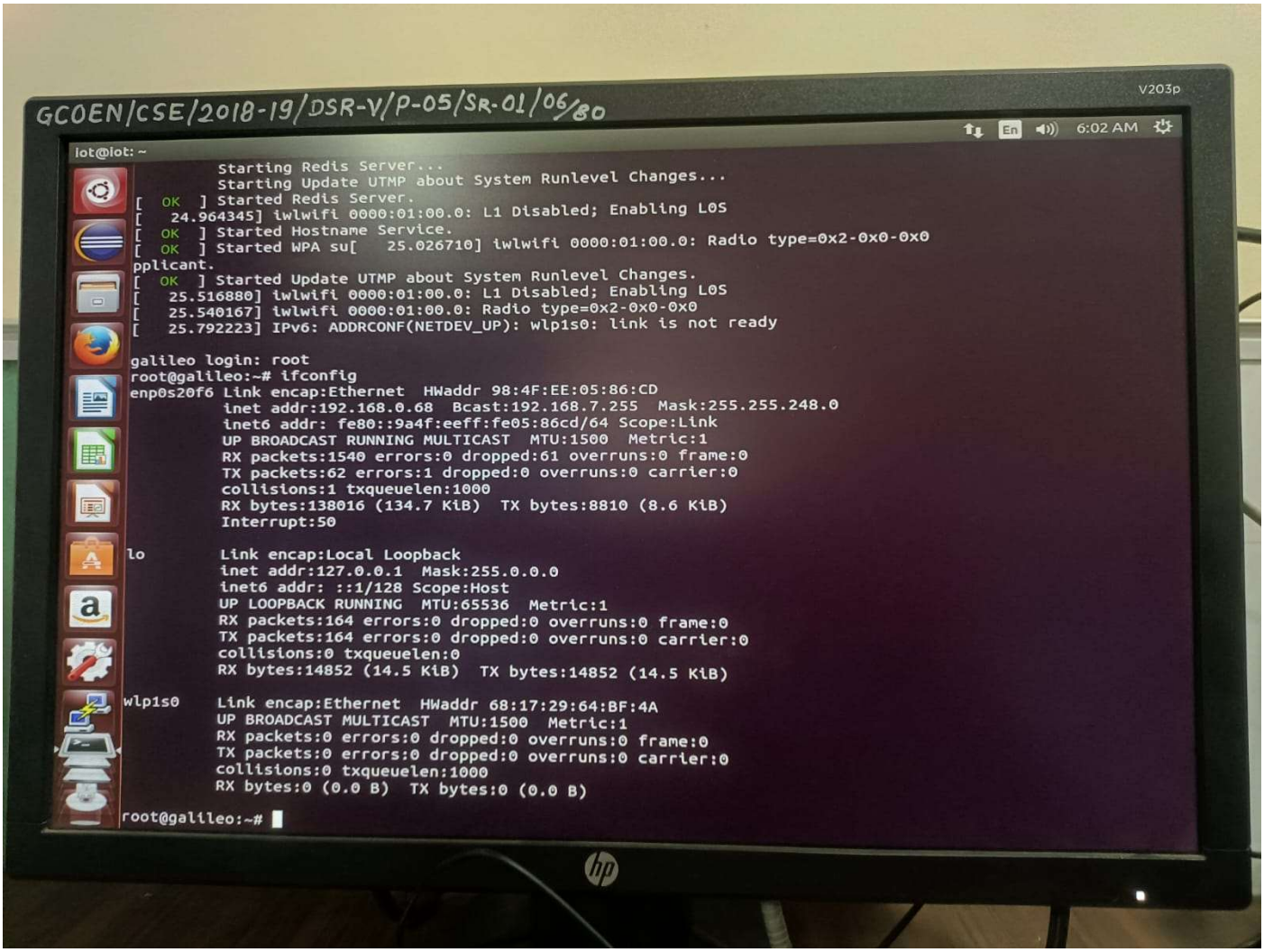
Steps to configure Intel Galileo board:

1) Assemble the hardware from IoT kit box intel / Galileo and connect Galileo board to the power through adaptor given.
2) Take out the SanDisk 32GB pen drive that is already loaded with Galileo Linux supportable OS username as IoT. Attach it to the computer, restart your computer as IoT and select pen drive in 1st position of boot order in BIOS settings.
3) Save the settings. Now OS will load into your computer, login by entering password (IOT123).
4) All necessary software are already installed in this OS, now go to the terminal, and write the following command. Make sure to attach Galileo to computer using SDTF to USB port cable.
5) Power on Galileo and write command on OS terminal: SUDO screen usbtty 020511.
6) This will load Galileo terminal on OS terminal.
7) Now Galileo will require password, enter “iot123”.
8) Now type Comment command in Galileo prompt.
9) Type “search wifi” in terminal by plugging “wifi receiver module” to computer, then you will be listed sum of SSID's of available wifi.
10) Copy ID of your wifi_ssid and type “add wifi wifi_ssid” and hit enter.
11) Now your Galileo board is also connected to internet exit by removing FTDI cable from computer.
12) Now all configurations have been done.

Program :-

None

Output :-



Conclusion :-

Study of IoT 3500 Development Kit and familiarization with Intel Galileo board and perform necessary software installations successfully.

Practical No :- 2

Aim :-

To interface LED with Intel Galileo board and write a program to turn on LED repeatedly after every one second interval.

Tool :-

Sr No.	Tools	Specifications	Qty
1	Development Kit	IoT 3500 Kit	1
2	Computer System	4GB RAM, 1TB HDD	1
3	LED Display	LD15, for Galileo	1

Theory :-

Steps to implement above statement in Galileo board:

1) Plug Galileo to computer and power on.
2) Power computer system enter passwords.
3) Open 'Eclipse IDE' on Linux OS that is already installed.
4) All the programs were given in workspace of Eclipse.
5) Click on create connection → new connection → enter Galileo IP address and name your connection.
6) Click Next → right click on connection → click start connection → connection will now change to green.
7) Write appropriate program and run by clicking on run button.
8) Write command to interface LED to Galileo
9) Upload the code to Galileo and click on run.

Program :- { Blinky.cpp - D3 }

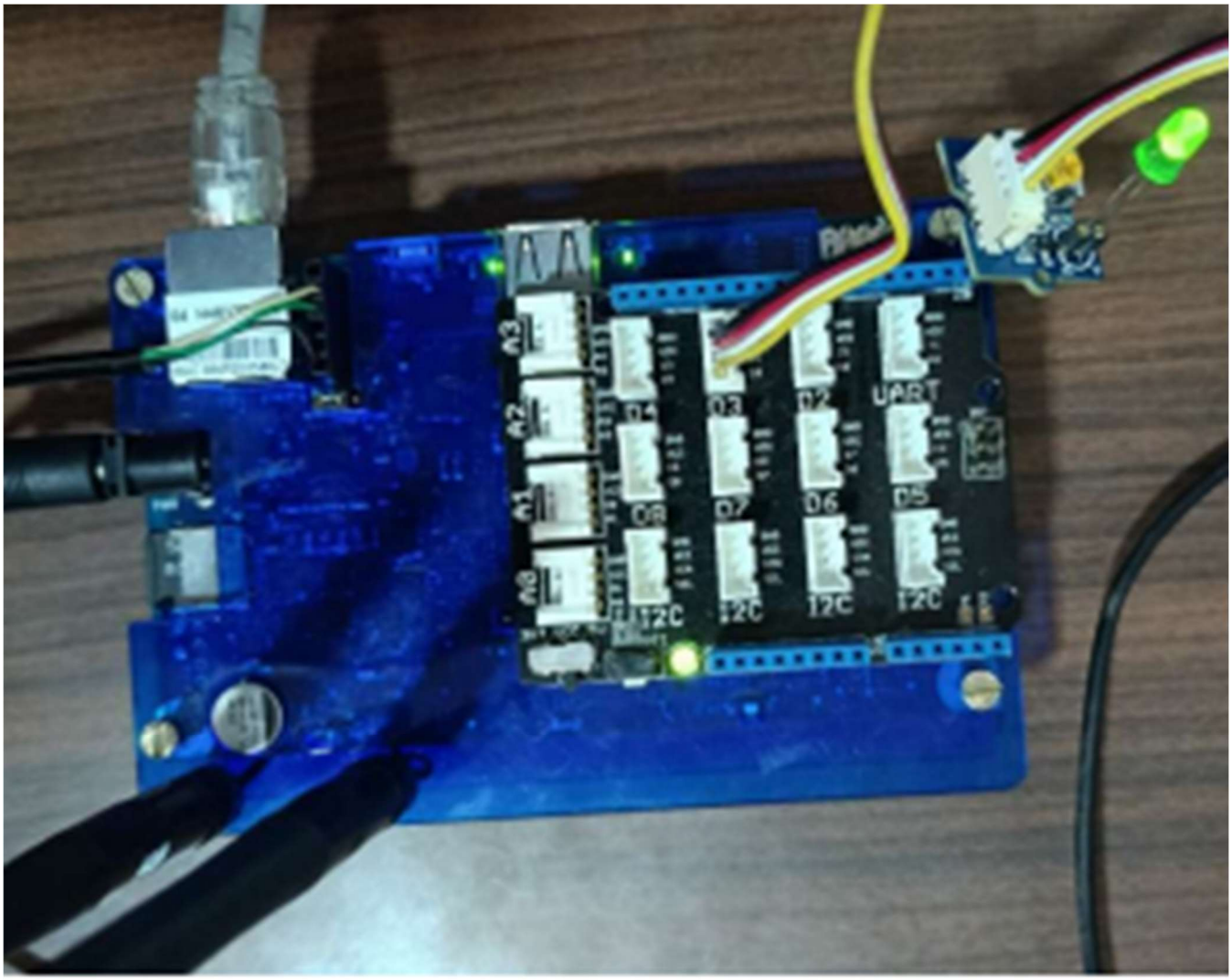
```
/**

* @file Blinky.cpp

* @brief  Demonstrate how to blink the on board LED by writing a digital value to an output pin using the MRAA library.

*

* @note No external hardware is needed for this experiment.

* @note Galileo has GPIO - Digital IO 13 connected to on board LED DS2.

* @note Additional linker flags: none.

*
```

```
* @see http://iotdk.intel.com/docs/master/mraa/
* @see http://download.intel.com/support/galileo/sb/galileo_boarduserguide_330237_001.pdf
*
* @bug No known bugs
**/
/* -- Includes -- */
/* MRAA library includes */

#include "mraa.h"

/*
 * for
 * mraa_get_platform_type()
 * mraa_gpio_init()
 * mraa_gpio_dir()
 * mraa_gpio_write()
 *
 */
/* Standard IO includes */

#include <stdio.h>

/*
 * for
 * fprintf()
 *
 */
/**
 * @brief Main function for LED blink on Galileo
 *
 * It performs following tasks:
 * 1. Initialize GPIO pin for Galileo using mraa_gpio_init() funtion.
 * 2. Configure the GPIO as a digital output.
 * 3. Enter loop to toggle the on board LED 10 times as below:
```

```
*                   Loop n times:
*                           Turn off LED
*                           wait 1 second
*                           Turn on LED
*                           wait 1 second
*                           Loop
*
* Example usage:
* @code{.c}
* int main(void)
* @endcode
*
* @param void This function does not take any input parameter.
* @return Returns MRA return code for Errors or Success.
*
* @see http://iotdk.intel.com/docs/master/mraa/gpio_8h.html
*
* @note For definition of MRAA return types, please refer types.h
* (under Includes/<MRAA include folder>).
*
* @warning None.
*/

int main()
{

        // create a GPIO object from MRAA using it

        mraa_gpio_context d_pin = NULL;
        d_pin = mraa_gpio_init(13);

        if (d_pin == NULL) { // error check
                fprintf(stderr, "MRAA couldn't initialize GPIO, exiting");
                return MRAA_ERROR_UNSPECIFIED;
```

```
        }

        // configure the GPIO pin as output

        if (mraa_gpio_dir(d_pin, MRAA_GPIO_OUT) != MRAA_SUCCESS) {
                fprintf(stderr, "Can't set digital pin as output, exiting");
                return MRAA_ERROR_UNSPECIFIED;
        };

        // loop to toggle on-board every second for 10 times

        for (int i=10;i>0;i--) {
                printf("LED OFF\n");
                mraa_gpio_write(d_pin, 0); // turn off LED
                sleep(1); //wait 1 second
                printf("LED ON\n");
                mraa_gpio_write(d_pin, 1); // turn on LED
                sleep(1); //wait 1 second
        }
        return MRAA_SUCCESS;
}
```

Output :-



Result :-

Compilation and Installation of LED interfacing after every one second is done and process understood.

Conclusion :-

Target board of such types can be used in low-cost system designs using very less amount of components and can be used for many user defined application.

Practical No :- 3

Aim :-

To interface push button with Intel Galileo board and write program to turn on LED when push button is pressed.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM, 1TB HDD	1
3	LED	LD15	1

Theory :-

Steps to interface push button:

Materials used:

1) Intel Galileo board
2) Push button
3) LED
4) Resistor
5) Connecting wires

Hardware setup steps:

1) Connect one terminal of push button to a digital input pin (i.e., D2) on Galileo board.
2) Connect jumper wired from D2 on Galileo board to some row on push button.
3) Connect LED to Galileo board using another connecting wire.
4) Power ON Galileo and connect with computer system.
5) Open eclipse and connect to your Galileo connection.
6) Write program in new file and save.
7) Upload into Galileo and click on run.

Program :- { (Button - D2 and Light - D3) Blinky.Cpp }

```
#include “mraa.h”
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#define BUTTON_PIN 2
```

```
#define LED_PIN 13
int main()
{
mraa.init();
mraa.gpio.context button=mraa.gpio init(BUTTON_PIN);
mraa_gpio_dir(button,MRAA_GPIO_IN);
mraa_gpio_context led = mraa_gpio_init(LED_PIN);
mraa_gpio_dir(led, MRAA_GPIO_OUT);
for (;;) {
        if (mraa_gpio_read((button)) mraa_gpio_write(led, 1);
        else mraa_gpio_write(led, 0);
        unsleep(100000);
}
mraa_gpio_close(button);
mraa_gpio_close(led);
mraa_deinit();
return 0;
}
```

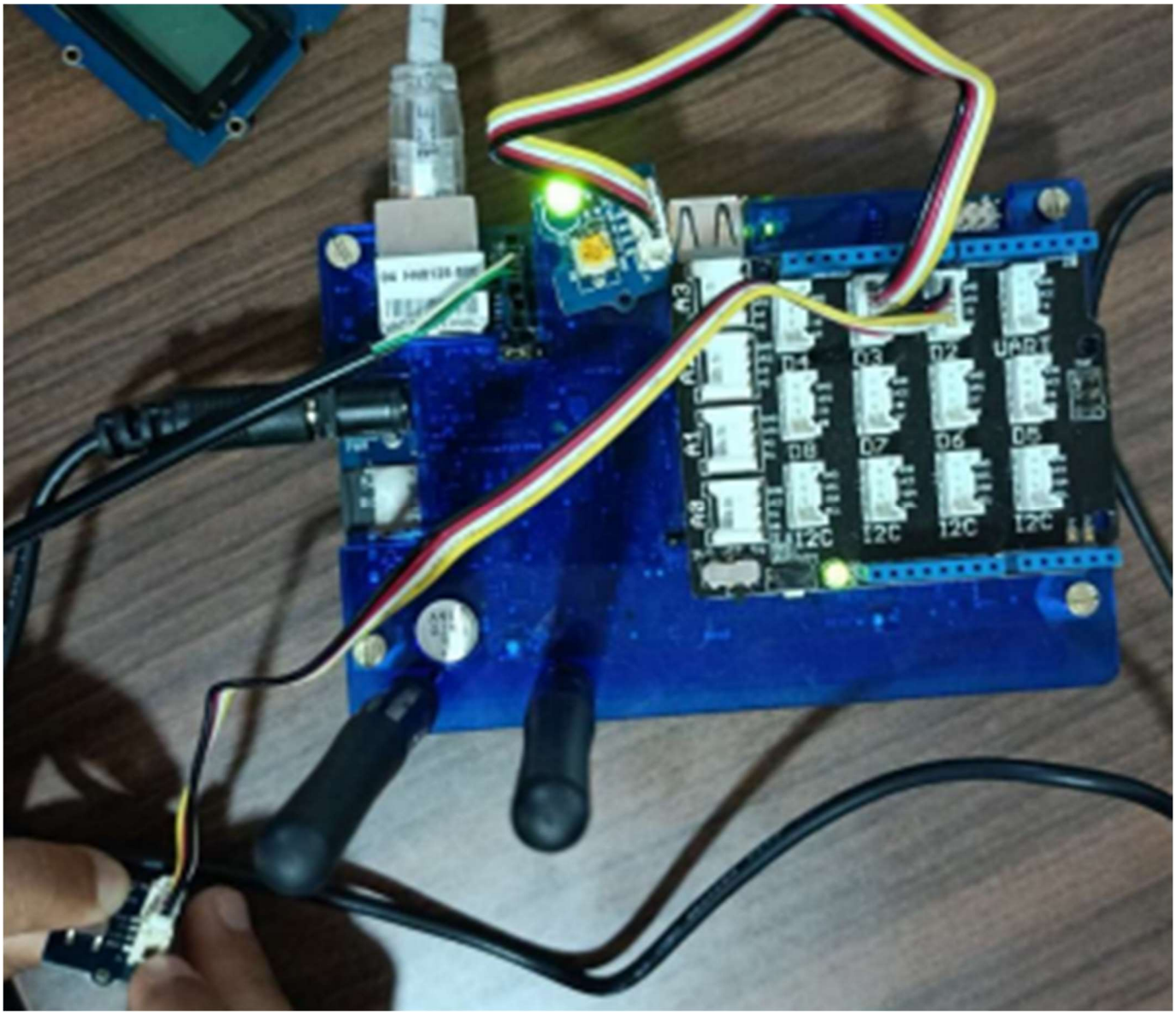
Program For Button :- { Button.cpp - D4 }

```
/**
* @file Button.cpp
* @brief Demonstrate how to use Grove button connected to Intel Galileo
*
* The Grove - Button is a momentary push button. It contains one independent
* "momentary on/off" button. 'Momentary' means that the button rebounds on its
*  own after it is released. The button outputs a HIGH signal when pressed,
*  and LOW when released.
*
* @note grovebutton sensor needs to be connected to the Galileo Arduino via base
* shield at Digital input D4.
* @note Additional linker flags: "upm-grove".
*
* @see http://iotdk.intel.com/docs/master/upm/
```

```
* @see https://software.intel.com/en-us/iot/hardware/sensors/grove-button
* @see http://www.seeedstudio.com/wiki/Grove_-_Button
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "grove.h" // for button->value()

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()

/**
 * @brief Main function for button example
 *
 * It performs following tasks:
 * 1. Creates an instance button of class grovebutton using UPM.
 * 2. Enter a loop to read and print button D4 values at intervals of 0.5 second.
 * 3. Exit loop after detecting button press for 5 times
 *
 * Example usage:
 * @code{.c}
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_grove_button.html
 *
```

```
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

int main()
{

   // Create the button object using GPIO pin D4

   upm::GroveButton* button = new upm::GroveButton(4);

   // Read the input and print, waiting one second between readings

   int count = 5;
   int button_val=0;
   while( count > 0 ) {
         button_val = button->value(); //read button value and store to local variable
         printf ("Program will exit after %d button presses\n", count);
         printf ("Button value is: %d\n ", button_val);
      if (button_val)
         count--;
      usleep(500000); //0.5 second
   }
         printf ("Exiting, bye!");

   // Delete the button object

   delete button;
}
```

Output:-



Result :-

Given problem statement successfully implemented.

Conclusion :-

Program to interface push button to LED to turn ON when button pressed implemented successfully.

Practical No :- 4

Aim :-

To interface buzzer with Intel Galileo board and write a program to buzz on and off with a delay of one second.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM,1TB HDD	1
3	Buzzer, Connecting wires	-	1
4	Eclipse IDE	-	1

Theory :-

Steps to interface buzzer with intel Galileo:

Materials used: Buzzer, Connecting wires, Galileo board.

Hardware setup:

1) Connect buzzer to Galileo board using connecting wire to digital input pin (i.e., D5).
2) Connect power adapter of board and turn it ON.
3) Open Eclipse IDE in computer system.
4) Connect to Galileo connection.
5) Write code in new file.
6) Upload code in Galileo using console.
7) Click on RUN button.

Program :- { Buzzer.cpp - D5 }

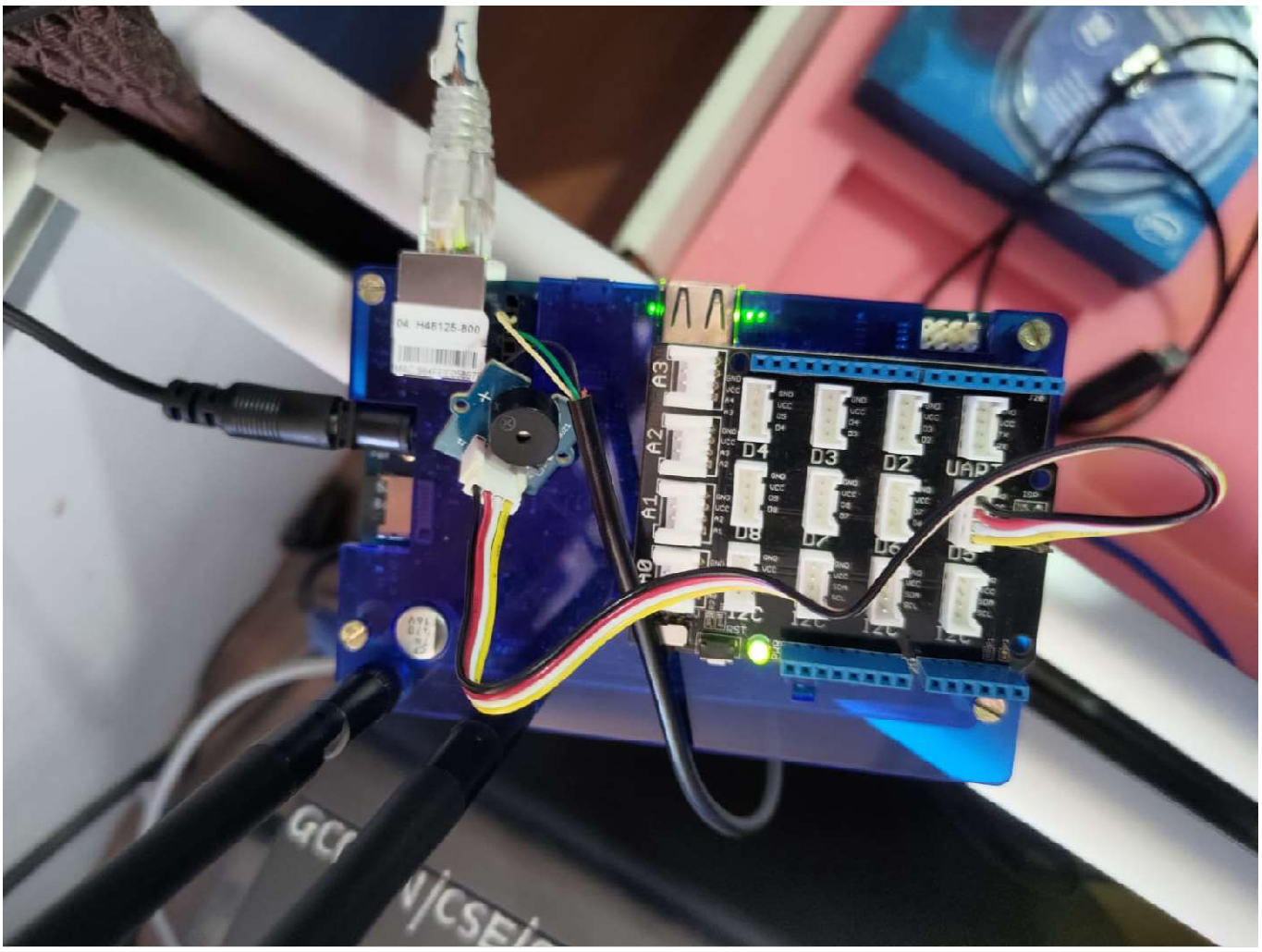
```
/**
* @file Buzzer.cpp
* @brief Demonstrate how to play tones using Buzzer connected to Intel Galileo board
*  using UPM library
*
* The Grove - Buzzer module has a piezo buzzer as the main component. The piezo
* can be connected to digital outputs, and will emit a tone when the output is
* HIGH. Alternatively, it can be connected to a pulse-width modulation
* output to generate various tones and effects.
```

```
*
* @note grovebuzzer sensor needs to be connected to the Galileo Arduino via base
* shield at D5.
* @note Additional linker flags: "upm-buzzer".
*
* @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_buzzer.html
* @see https://software.intel.com/en-us/iot/hardware/sensors/grove-buzzer
* @see http://www.seeedstudio.com/wiki/Grove_-_Buzzer
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include <buzzer.hpp>

/* for
 * sound->playSound()
 * sound->stopSound()
 * sound->setVolume()
 * sound->getVolume()
 */
/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()

/**
 * @brief Main function for Buzzer example
 *
 * It performs following tasks:
 * 1. Creates an instance button of class Buzzer using UPM.
 * 2. play sound (DO, RE, MI, etc...) for 1 second, pausing for 0.1 seconds between notes.
```

```
 *
 * Example usage:
 * @code{.c}
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_buzzer.html#details
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

int main()
{
   int chord[] = { DO, RE, MI, FA, SOL, LA, SI, DO, SI };

            // create Buzzer instance

            upm::Buzzer* sound = new upm::Buzzer(5);
            printf("Volume = %f\n", sound->getVolume());
            sound->setVolume(0.6);
            printf("Volume = %f\n", sound->getVolume());
            fflush(stdout);

            // play sound (DO, RE, MI, etc...), pausing for 0.1 seconds between notes

            printf("\nPlaying notes, pausing for 0.1 seconds between notes...\n");
            fflush(stdout);
            for (int chord_ind = 0; chord_ind < 7; chord_ind++) {
```

```
            // play each note for one second

            printf(" %d\n",  sound->playSound(chord[chord_ind], 500000) );
            usleep(100000);
        }
        printf("Exiting, bbye!\n");
        delete sound;
}
```

Output :-



Result :-

Program implemented correctly, and problem understood clearly.

Conclusion :-

Program to interface buzzer, with Intel Galileo board to buzz ON and OFF after delay of one second implemented successfully.

Practical No :- 5

Aim :-

To interface temperature sensor with intel Galileo board and write a program to print temperature readings.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM, 1TB HDD	1
3	Temperature Sensor	-	1

Theory :-

Steps to interface temperature sensor with intel Galileo board:

Materials used of kit: Temperature Sensor, Connecting Wires, Galileo board.

Hardware / Software Setup:

1) Connect temperature sensor to Galileo board by using connecting wires to input pin (e.g., A0) to read analog values.
2) Power ON Galileo board, connect it to computer system.
3) Open Ellipse IDE in computer, connect to connection write code in new file.
4) Mention temperature sensor pin no. on program.
5) Upload the code to Galileo board.
6) Click on Run.

Program :- { Temperature.cpp - A0 }

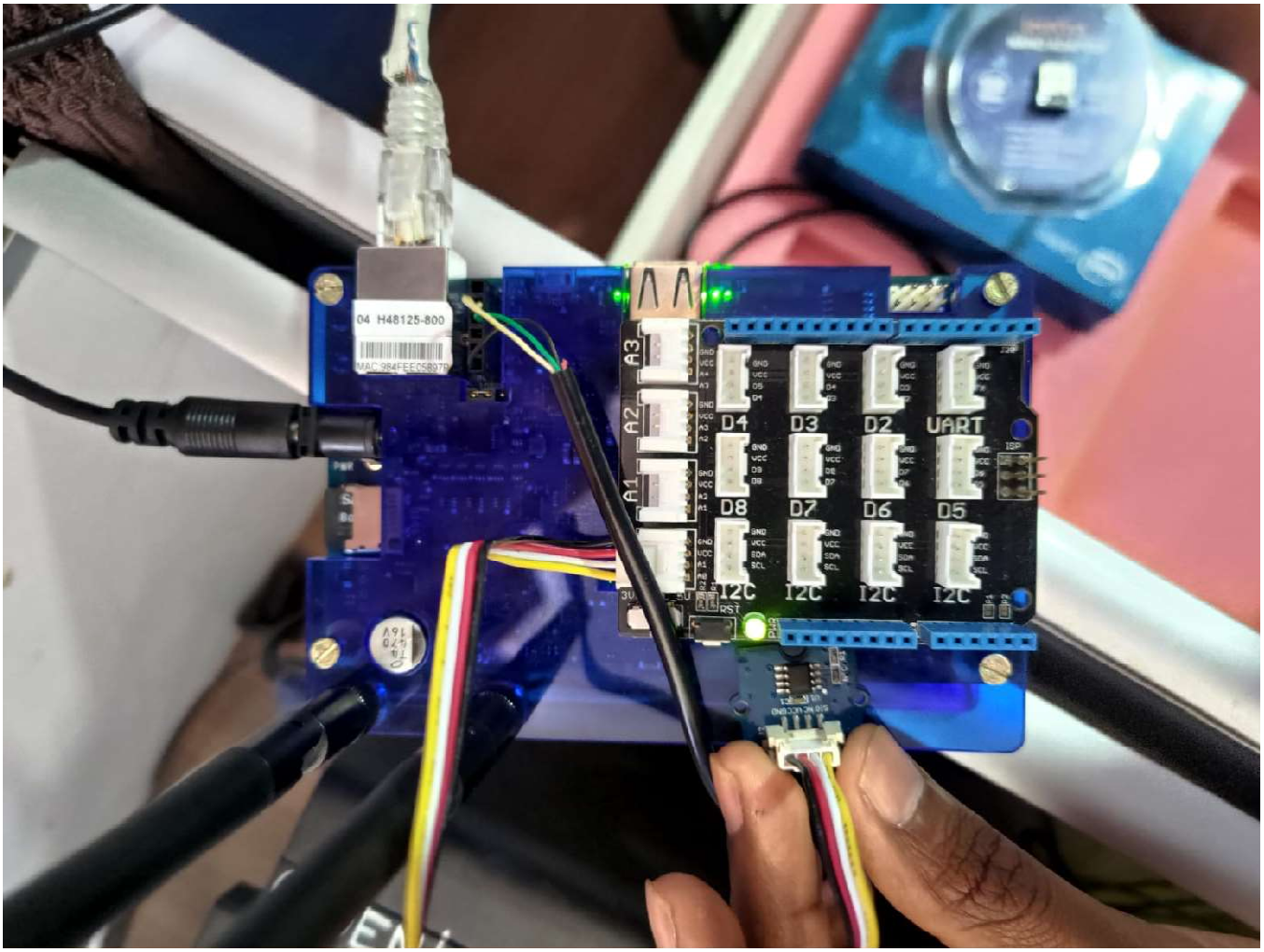
```
/**
* @file Temperature.cpp
* @brief  Demonstrate how to acquire Temperature values from the Grove Temperature sensor
*  connected to Intel Galileo board using UPM library
*
* The Grove - Temperature Sensor uses a Thermistor to detect the ambient temperature.
* The resistance of a thermistor will increase when the ambient temperature decreases.
* It's this characteristic that we use to calculate the ambient temperature.
* The detectable range of this sensor is -40 - 125ºC, and the accuracy is ±1.5ºC
*
* @note Grove Temperature sensor needs to be connected to the Galileo Arduino via base
* shield Analog IO pin A0
```

```
*
* @note Default reference voltage for ADC is 5V, so full scale deflection is at 5V
* @note Ensure base shield VCC switch is set to 5V for this experiment,
* hence 5V input = 1024 (10 bit)
*
* @see http://iotdk.intel.com/docs/master/mraa/aio_8h.html
* @see http://www.seeedstudio.com/wiki/Grove_-_Temperature_Sensor_V1.2
* @see http://www.seeedstudio.com/wiki/images/4/4f/Grove_-_Temperature_sensor_v1.1.pdf
* @see http://www.seeedstudio.com/wiki/images/a/a1/NCP18WF104F03RC.pdf
* @bug No known bugs
*
*/
/* -- Includes -- */

#include "mraa/aio.h" //for mraa_aio_read()
#include <math.h> // for math functions

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()
#include "jhd1313m1.h"
#include "grove.h"

/**
 * @brief Main function for temperature sensor example
 *
 * It performs following tasks:
 * 1. Create object of class analog input and initialize A0 pin for input
 * 2. Enter a loop to acquire analog input and convert to temperature values
 * then print 5 times at time intervals of 1 second.
 *
 * Example usage:
 * @code{.c}
```

```
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/mraa/aio_8h.html
 *
 * @warning None.
 */

int main()
{
   mraa_aio_context adc_a0;
   uint16_t adc_value = 0;
   const int B=4275;               // B value of the thermistor
   const int R0 = 100000;          // R0 = 100k

   //create object of analog input class using mraa lib

   adc_a0 = mraa_aio_init(0);
   if (adc_a0 == NULL) {
      return 1;
   }
   for (int i=10; i>0;i--) {
      adc_value = mraa_aio_read(adc_a0); //Max value @ 5V = 1024
      printf("ADC A0 read value : %d\n", adc_value);
      float R = 1023.0/((float)adc_value)-1.0;
      R = 100000.0*R;
      float temperature=1.0/(log(R/100000.0)/B+1/298.15)-273.15;//convert to temperature as per datasheet ;
      printf("Temperature value : %.2f Degree Celsius\n", temperature);
      sleep(1);
   }
   mraa_aio_close(adc_a0);
```

```
    printf("Exiting .. Bbye!");
    return MRAA_SUCCESS;
}
```

Output :-



Conclusion :-

Program to interface temperature sensor with Galileo board to read temperature and print it implemented successfully.

Practical No :- 6

Aim :-

To interface touch sensor with intel Galileo board and write a program to sense a finger when it is placed on intel Galileo board.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM, 1TB HDD	1

Theory :-

Steps to interface touch sensor with intel Galileo board

Materials used of kit: Touch Sensor, Connecting Wires, Galileo board.

Hardware / Software setup:

1) Connect touch sensor to Galileo board by using connecting wires to input analog pin of Galileo board (e.g., D4).
2) Power ON Galileo board, connect it to computer system.
3) Open Ellipse IDE on computer, connect to existing connection.
4) Write program on new file.
5) Save the program.
6) Upload to Galileo board using console command.
7) Install related pkg

   Enter pkg update
   pkg install i2C-tools
   i2_detect-x-y < i2C_bus_number >

8) Click on RUN.
9) Check the output on Galileo board.

Program :- { TouchInterrupt.cpp – D4 }

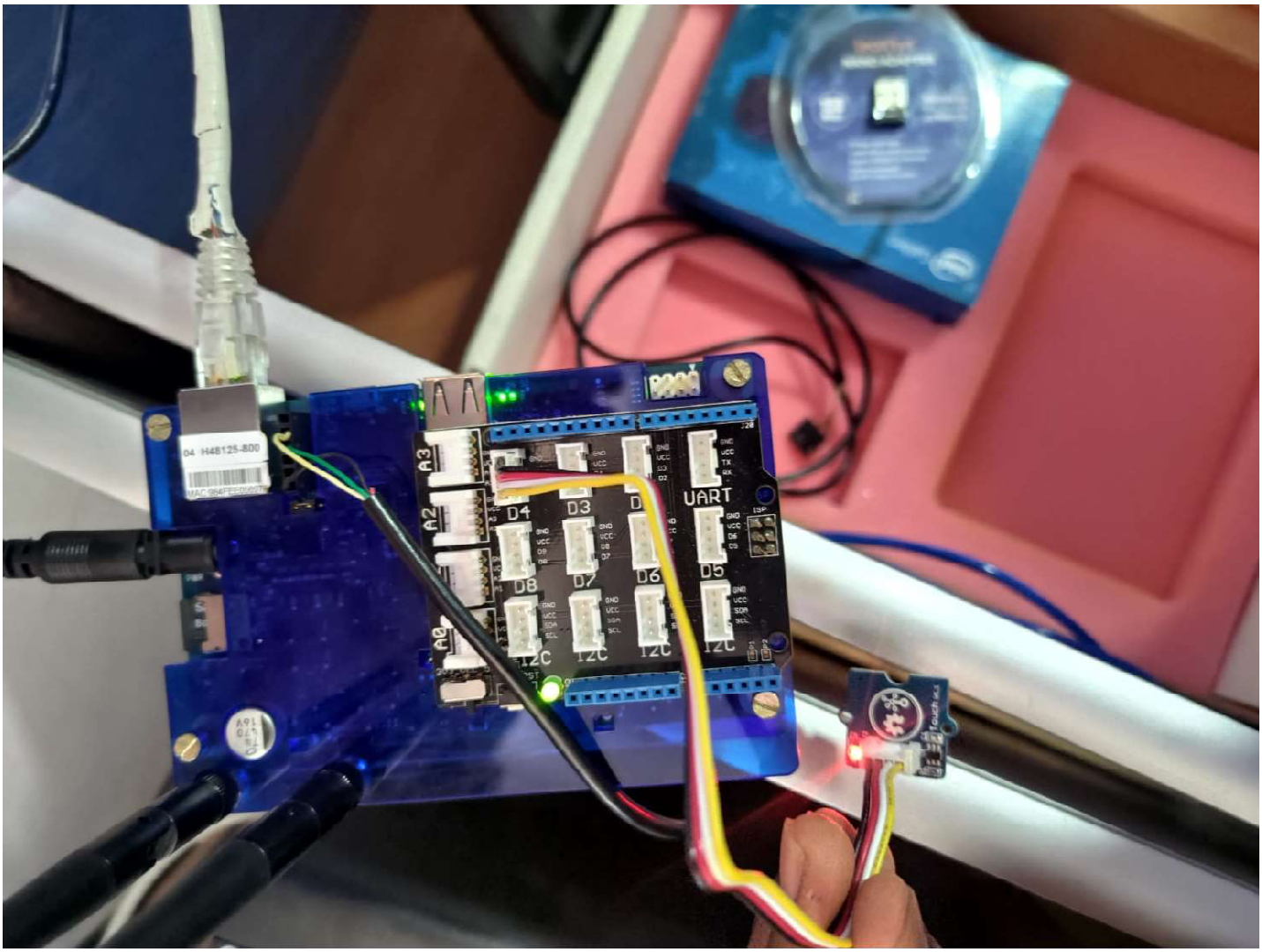
```
/**
* @file TouchInterrupt.cpp – D4
* @brief Demonstrate how to trigger interrupt using Grove button
*
* This touch sensor detects when a finger is near the metallic pad by the change
* in capacitance. It can replace a more traditional push button. The touch sensor
```

```
* can still function when placed under a non-metallic surface like glass or plastic.
*
* @note grovetouch sensor needs to be connected to the Galileo Arduino via base
* shield D4.
* @note Additional linker flags: "upm-ttp223".
*
* @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_t_t_p223.html
* @see https://software.intel.com/en-us/iot/hardware/sensors/ttp223-touch-sensor
* @see http://www.seeedstudio.com/wiki/Grove_-_Touch_Sensor
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "ttp223.h" // for button->value()

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()
void touchISR (void*);
int count = 5;

/**
 * @brief touchISR is the Interrupt Service Routine for touch event
 *
 * It performs following tasks:
 * 1. It gets called upon touch event
 * 2. Update the touch event count and print a message
 * 3. Exit
 *
 * Example usage: ISR will be called when events occur, it has to be registered
```

```
 * for touch event by another function (main)
 * @code{.c}
 * touch->installISR(mraa::EDGE_FALLING, &touchISR, NULL);
 * @endcode
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_t_t_p223.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 *
 */

void touchISR(void*)
{
        count--;
        printf("\nHello World from ISR, will exit after %d touch events", count);
        fflush(stdout);
}

/**
 * @brief Main function for touch sensor example
 *
 * It performs following tasks:
 * 1. Creates an instance button of class TTP223 using UPM.
 * 2. Installs an ISR function which will be called on touch input.
 * 3. ISR prints message on touch event.
 *
 * Example usage:
 * @code{.c}
 * int main(void)
 * @endcode
```

```
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_t_t_p223.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 *
 */

int main()
{

   // Create the button object using GPIO pin 4

   upm::TTP223* touch = new upm::TTP223(4);

   // Read the input and print, waiting one second between readings

   touch->installISR(mraa::EDGE_FALLING, &touchISR, NULL);
         printf("\nWelcome, waiting for touch event.\nWill exit after 5 events");
         fflush(stdout);
   while(count>0);
   printf("\nExiting .. Bbye!");
   delete touch;     // Delete the button object
}
```

Output :-



Conclusion :-

Program to interface touch sensor with Galileo board and sense finger, if it is place on Galileo board implemented successfully.

Practical No :- 7

Aim :-

To interface light sensor with Galileo board and write a program to detect surrounding light intensity.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM, 1TB HDD	1
3	Light sensor	LD15	1

Theory :-

Steps to interface light sensor to Galileo.

Materials required: Light sensor, Galileo board, Connecting wires.

Hardware Setup:

1) Connect light sensor to Galileo board using connecting wire to pin Input analog e.g., A0.
2) Power ON Galileo board. Connect it to computer system using FTDI to USB cable.
3) Open Eclipse IDE in computer, connect to existing connection user.
4) Write a program in new file of eclipse. Upload it to Galileo board after interfacing light sensor using console command.
5) Save file, click on RUN.
6) Measure output in console window.

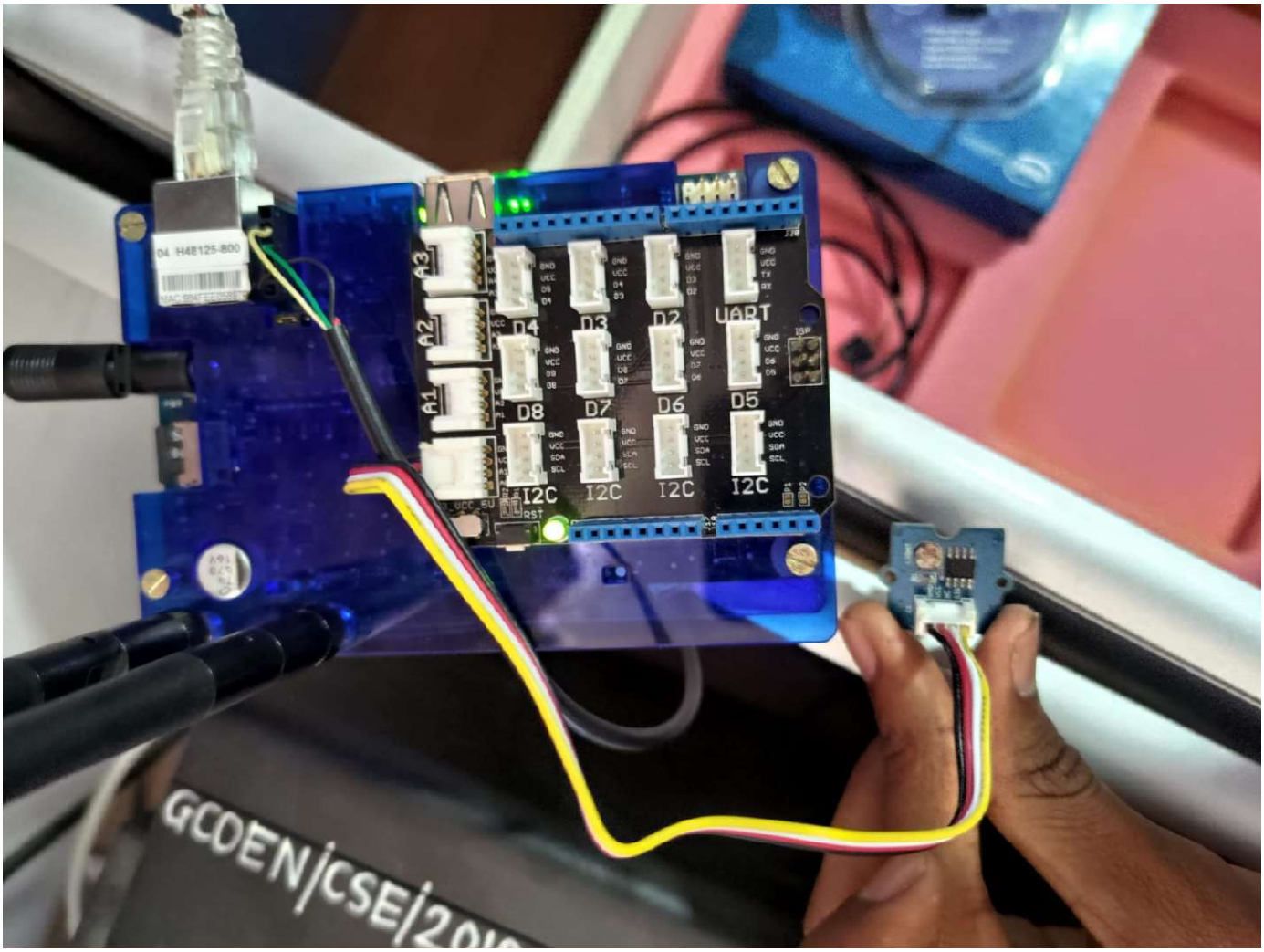
Program :- { Light.cpp - A0 }

```
/**
* @file Light.cpp
* @brief Demonstrate how to measure light using sensor connected to
* Intel Galileo board using UPM library
*
* @note grovelight sensor needs to be connected to the Galileo Arduino via base
* shield analog port A0.
* @note Additional linker flags: "upm-grove".
*
* @see http://iotdk.intel.com/docs/master/upm/
* @see https://software.intel.com/en-us/iot/hardware/sensors/grove-light-sensor
* @see http://www.seeedstudio.com/wiki/Grove_-_Light_Sensor
```

```
* @see http://akizukidenshi.com/download/ds/senba/GL55%20Series%20Photoresistor.pdf
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "grove.h" //for light->name(), light->value()

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()

/**
* @brief Main function for light sensor example
*
* It performs following tasks:
* 1. Creates an instance light of class grovelight using UPM.
* 2. Enter a loop to acquire and print light values at intervals of 1 second.
* 3. Exit loop after 5 iterations.
*
* Example usage:
* @code{.c}
* int main(void)
* @endcode
*
* @param void This function does not take any input parameter.
* @return Returns 0.
*
* @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_grove_light.html
*
* @note UPM library has dependency on MRAA library functions.
```

```
*
* @warning None.
*/

int main()
{

        // Create the light sensor object using AIO pin 0

        upm::GroveLight* light = new upm::GroveLight(0);

        // Read the input and print both the raw value and a rough lux value,
        // waiting one second between readings

        for (int i=20;i>0;i--) {
                printf(" Light value is %f which is roughly %d lux \n", light->raw_value(), light->value());
                fflush(stdout);
                sleep(1);
        }

        // Delete the light sensor object

        printf("Exiting .. bbye!");
        delete light;
}
```

Output :-



Conclusion :-

Program to interface light sensor with Galileo board and detect surrounding light intensity implemented successfully.

Practical No :- 8

Aim :-

To interface sound sensor with Intel Galileo board and write a program to detect surrounding sound intensity.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM,1TB HDD	1
3	Sound sensor	-	1
4	Connecting wires	-	1

Theory :-

Steps to interface sound sensor to Galileo :

Materials used: Sound sensor, Galileo board, Connecting wires.

Hardware Setup:

1) Connect sound sensor to Galileo board using connecting wire to analog input pin. (e.g., A0)
2) Power on Galileo board. Connect it to computer system using FTDI to USB cable.
3) Open Eclipse workspace in computer, by attaching OS pen drive given in development kit.
4) Write program in new file. Connect to Galileo connection and save file.
5) Upload program to Galileo board using console command.
6) Click on RUN button of IDE.
7) Measure output on console window.

Program :- { Mic.cpp – A0 }

```
/**
* @file Mic.cpp
* @brief Demonstrate how to use Grove Mic connected to Intel Galileo
*
* Grove - Sound Sensor can detect the sound strength of the environment. The main
* component of the module is a simple microphone, which is based on the LM358 amplifier
* and an electret microphone. This module's output is analog and can be easily sampled
*
```

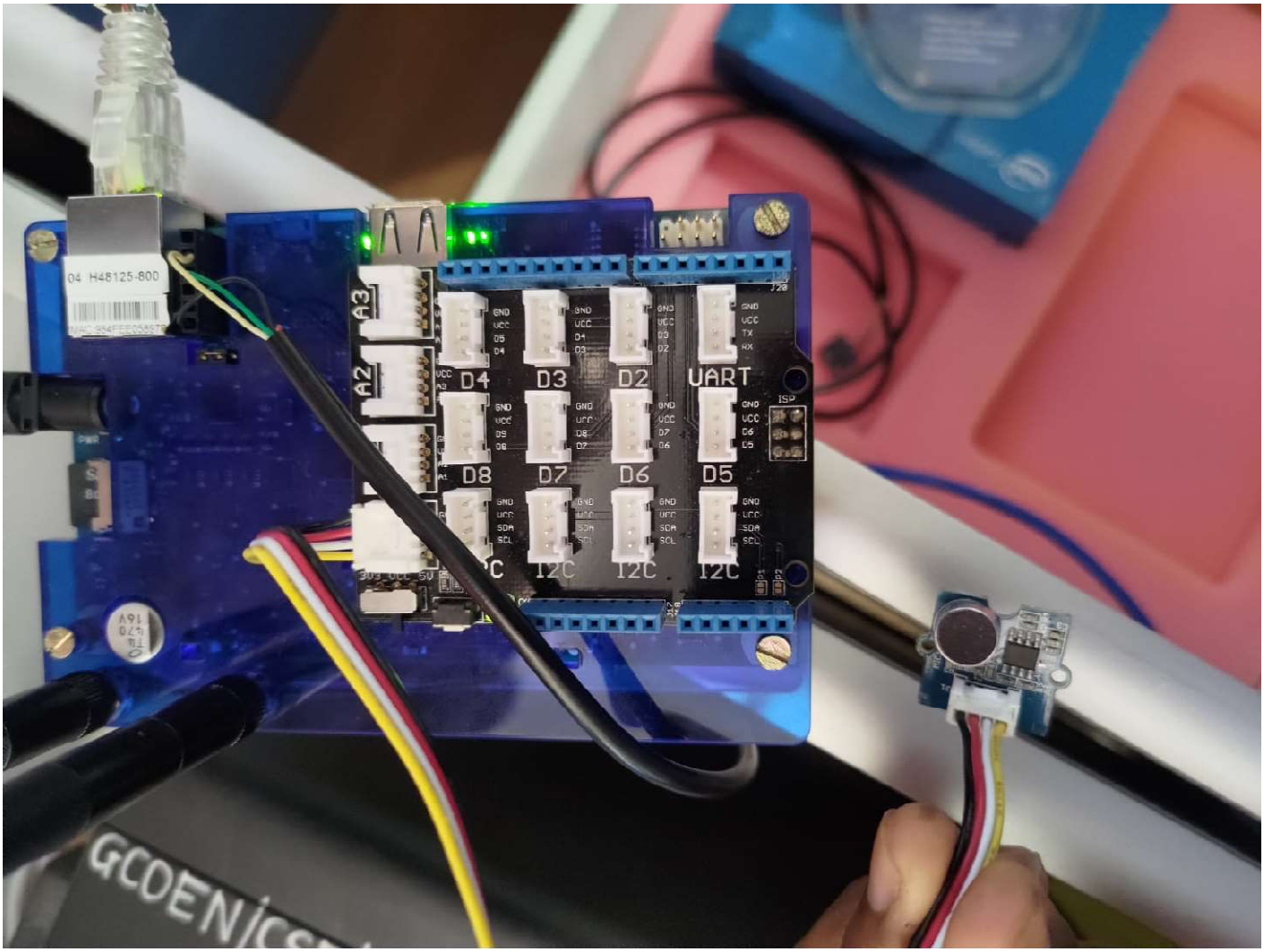
```
* @note grove mic sensor needs to be connected to the Galileo Arduino A0 via base
* shield.
* @note Additional linker flags: "upm-mic".
*
* @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_microphone.html#details
* @see http://www.seeedstudio.com/wiki/Grove_-_Sound_Sensor
* @see https://software.intel.com/en-us/iot/hardware/sensors/analog-microphone
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "mic.h" // for

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()
#include <signal.h>
#include <sys/time.h>
int is_running = 1;
uint16_t buffer [128]; //define buffer to store captures values
upm::Microphone *mic = NULL;      //create microphone object

/**
 * @brief sig_handler is the routine to catch kill signal events
 *
 * It performs following tasks:
 * 1. It gets called upon kill signal received (Ctrl + C to stop program)
 * 2. Set flag to gracefully exit the program
 * 3. Exit
 *
```

```
 * Example usage: sig_handler() has to be installed to capture signal event
 * in another function (main)
 * @code{.c}
 * signal(SIGINT, sig_handler);
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_grove_button.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

void sig_handler(int signo)
{
   printf("got signal\n");
   if (signo == SIGINT) {
      is_running = 0;
   }
}

/**
 * @brief Main function for sound sensor example
 *
 * It performs following tasks:
 * 1. Install signal handler to catch kill signal
 * 2. Creates an instance Mic of class Microphone using UPM.
 * 3. Take a sample every 2 microseconds; find the average of 128 samples
 * 4. Print a running graph of the averages
 *
 * Example usage:
```

```
 * @code{.c}
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_microphone.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

int main(int argc, char **argv)
{

    // Attach microphone to analog port A0

    mic = new upm::Microphone(0);
    if (signal(SIGINT, sig_handler) == SIG_ERR)
            printf("\ncan't catch SIGINT\n");
    thresholdContext ctx;
    ctx.averageReading = 0;
    ctx.runningAverage = 0;
    ctx.averagedOver   = 2;

    // Infinite loop, ends when program is cancelled
    // Repeatedly, take a sample every 2 microseconds;
    // find the average of 128 samples; and
    // print a running graph of the averages

    while (is_running) {
        int len = mic->getSampledWindow (2, 128, buffer);
```

```
        if (len) {
            int thresh = mic->findThreshold (&ctx, 30, buffer, len);
            mic->printGraph(&ctx);
            if (thresh) {
                // do something ....
            }
        }
    }
    printf ("exiting application\n");
    delete mic;
    return 0;

}
```

Output :-



Conclusion :-

Program to interface sound sensor with Galileo board and detecting surround sound implemented successfully.

Practical No :- 9

Aim :-

To interface LCD with intel Galileo board and write a program to display any text.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM,1TB HDD	1
3	LCD Display	-	1
4	Jumper Wires	-	1

Theory :-

Steps to interface the LCD display and write a program to display text.

1) Connect the VCC pin of the LCD display to the VCC pin on the Galileo board.
2) Connect the GND pin of the LCD display to the GND pin on the Galileo board.
3) Connect the SDA pin of the LCD display to the SDA pin on the Galileo board.
4) Connect the SCL pin of the LCD display to the SCL pin on the Galileo board.
5) Connect the Galileo board to a computers and open Arduino IDE.
6) Go to “Sketeli” -> “Include Library” -> “Manage libraries”.
7) In the library Manager. Search for “liquid crystal 12C”.
8) Install the “liquid crystal 12C” library by Arduino.

Program :- { LCD.cpp - last I2C which is close to D5 }

```
/**
* @file LCD.cpp - last I2C which is close to D5
* @brief  Demonstrate how to use the 16x2 LCD with RGB backlight connected to
*  Intel Galileo board using UPM library
*
* LCD module used here is JHD1313M1 which has an I2C controller for the 16x2
* LCD and an RGB backlight LED controller PCA9633 also I2C based. Hence the
* module has two I2C addresses: one belongs to LCD controller that controls the
* HD44780-based display, and the other controls only the backlight.
*
* @note Grove LCD RGB Backlight module needs to be connected to Galileo Arduino
*  via base shield's I2C port.
* @note Grove LCD needs 5V to operate normally, ensure Grove shield switch
```

```
* is set to 5V
* @note Galileo has an exposed I2C bus = 0 and I2C addresses:
* LCD controller = 0x3E ; RGB LED = 0x62.
* @note Additional linker flags: "upm-i2clcd".
*
* @see http://iotdk.intel.com/docs/master/upm/
* @see https://software.intel.com/en-us/iot/hardware/sensors/jhd1313m1-display
* @see http://www.seeedstudio.com/wiki/Grove_-_LCD_RGB_Backlight
* @see http://www.seeedstudio.com/wiki/images/0/03/JHD1214Y_YG_1.0.pdf
* @see http://www.seeedstudio.com/wiki/images/1/1c/PCA9633.pdf
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "jhd1313m1.h"

/*for
        lcd->setCursor();
   lcd->write();
   lcd->setCursor();
   lcd->setColor();
 */
/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()



/**
 * @brief Main function for printing text on LCD and changing backlight colors
 *
```

```
 * It performs following tasks:
 * 1. Creates an instance lcd of class Jhd1313m1 using UPM.
 * 2. Set cursor position
 * 3. Print text
 * 4. Enter loop to toggle backlight color between Red->Green->Blue at
 * intervals of 1 second.
 *
 * Example usage:
 * @code{.c}
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_jhd1313m1.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

int main(void)
{

   // 0x62 RGB_ADDRESS, 0x3E LCD_ADDRESS

         upm::Jhd1313m1 *lcd;
   lcd = new upm::Jhd1313m1(0, 0x3E, 0x62); //Create lcd instance

   //arguments: I2C addresses of LCD controller and LED backlight controller

   printf("Display text on LCD\n");
   lcd->setCursor(0,0);          //bring cursor to top left corner
```

```
    lcd->write("Batch 4");        //print text
    lcd->setCursor(1,2);          //bring cursor to second row
    lcd->write("welcome !");    //print text
    printf("Sleeping for 5 seconds\n");
    sleep(5);
    printf("Starting Color loop...\n");

    //Run loop for toggling backlight color between Red->Green->Blue x 5 times

    for (int i = 5; i>0 ;i--){
    lcd->setColor(255,220,220);          //set backlight color to Red
    sleep(1);
    lcd->setColor(0,255,0);      //set backlight color to Green
    sleep(1);
    lcd->setColor(0,0,125);      //set backlight color to Blue
    sleep(1);
    }
    printf("Exiting .. bbye!\n");
    delete lcd;    //free up memory.
    return 0;
}
```

Output :-

Result :-

After uploading the process is complete. The LCD display should start showing the message.

Conclusion :-

To interface LCD Display with intel Galileo board and write a program to display some text has been implemented successfully.

Practical No :- 10

Aim :-

To interface rotary angle sensor/ potentiometer with the Galileo board and write a program to control the readings of sensor of changing the angle of potentiometer.

Tool :-

Sr.no.	Tools	Specifications	Qty
1	IoT 3500 Development Kit	Intel Galileo kit	1
2	Computer System	4GB RAM,1TB HDD	1
3	Rotary Angle Senser	-	1
4	Jumper Wires	-	1

Theory :-

Steps to interface rotary angle senser with the Galileo board and write a program to control the readings of the sensor are as follows:

1) Connect the Grove Rotary Angle Sensor to an analog port (A1) on the shield. Then, connect the shield to the Galileo board.
2) Open Arduino IDE. Go to sketch -> Include library -> Manage libraries. Search for “UPM” for install the “GroveUPM” library by Arduino.
3) Copy the provided program code into a new sketch in the Arduino IDE.
4) Go to Tools -> Board and select “Galileo” or “Galileo Gen2” depending on your board. Then go to Tools -> Port and Select the COM port your Galileo board is connected to.
5) Click the upload button to upload the program to your Galileo board.
6) Open the serial monitor by going to Tools -> Serial Monitor. Set band rate to 9600 (usually the default).
7) Turn the knob of the rotary sensor. You should see readings on the serial monitor that update every second.

Program :- {RotaryAngle.cpp - A1}

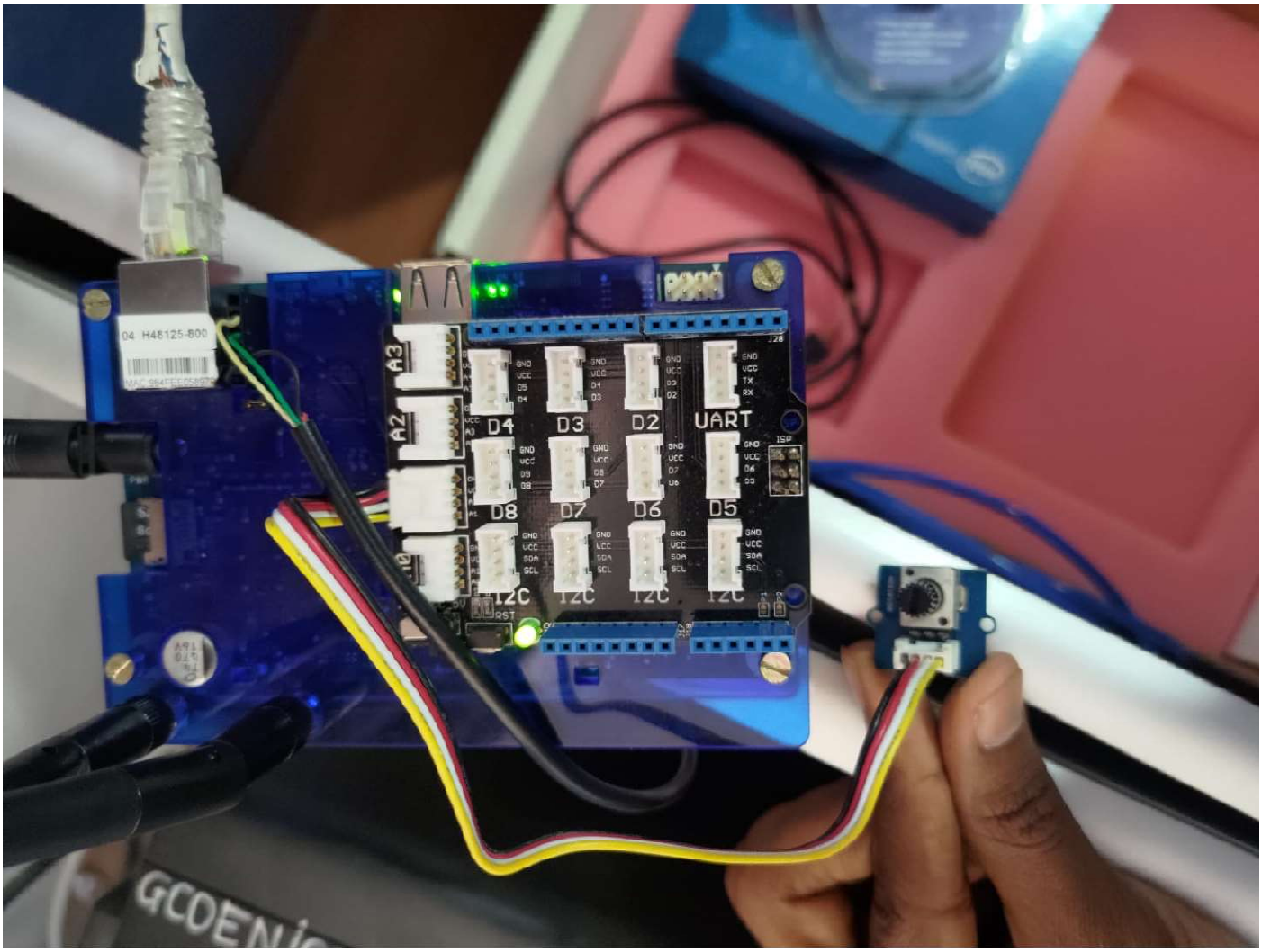
```
/**

* @file RotaryAngle.cpp

* @brief Demonstrate how to use Grove Rotary Angle Sensor connected to Intel Galileo

*

* The rotary angle sensor produces analog output between 0 and Vcc (5V DC)

* on its D1 connector. The D2 connector is not used. The angular range is 300 degrees

* with a linear change in value. The resistance value is 10k ohms.

* This may also be known as a 'potentiometer'

*
```

```
* @note grovebutton sensor needs to be connected to the Galileo Arduino A1 via base
* shield.
* @note Additional linker flags: "upm-grove".
*
* @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_grove_rotary.html
* @see http://www.seeedstudio.com/wiki/Grove_-_Rotary_Angle_Sensor
* @see http://www.csl.mtu.edu/cs4411.choi/www/Resource/signal.pdf
*
* @bug No known bugs
*
*/
/* -- Includes -- */
/* UPM library includes */

#include "grove.h" // for button->value()

/* Standard IO includes */

#include <stdio.h> // for printf()
#include <unistd.h> //for sleep()

/**
 * @brief Main function for light sensor example
 *
 * It performs following tasks:
 * 1. Creates an instance knob of class groverotary using UPM.
 * 2. Enter a loop to check for button value at intervals of 1 second.
 *
 * Example usage:
 * @code{.c}
 * int main(void)
 * @endcode
 *
 * @param void This function does not take any input parameter.
```

```
 * @return Returns 0.
 *
 * @see http://iotdk.intel.com/docs/master/upm/classupm_1_1_grove_button.html
 *
 * @note UPM library has dependency on MRAA library functions.
 *
 * @warning None.
 */

int main()
{
        // Instantiate a rotary sensor on analog pin A1

          upm::GroveRotary* knob = new upm::GroveRotary(1);

  // Read the input and print, waiting one second between readings

  while( 1 ) {
      float abs_value = knob->abs_value(); // Absolute raw value
      float abs_deg = knob->abs_deg();     // Absolute degrees
      float abs_rad = knob->abs_rad();     // Absolute radians
      float rel_value = knob->rel_value(); // Relative raw value
      float rel_deg = knob->rel_deg();     // Relative degrees
      float rel_rad = knob->rel_rad();     // Relative radians
      printf("Absolute: %4d raw %5.2f deg = %3.2f rad Relative: %4d raw %5.2f deg %3.2f rad\n",
          (int16_t)abs_value, abs_deg, abs_rad, (int16_t)rel_value, rel_deg, rel_rad);
      sleep(1); // Sleep for 1s
    }

  // Delete the button object

  delete knob;

  //Will the program ever reach here? If no, then how can we delete knob?
```

```
  //Hint: See boiler plate for link to signal document


}
```

Output :-



Result :-

When we run the program and rotate the knob of the rotary sensor, we see output on the serial monitor that updates every second.

Conclusion :-

The practical successfully demonstrates how to interface a rotary angle sensor (or potentiometer) with a Galileo board. The provided program reads the sensor values and displays them on the serial monitor allowing to observe the relationship between the sensors physical rotation and output readings.