**Faculty of Engineering**

**Ain Shams University**

Computer Engineering and Software Systems



Light Fidelity (Li-Fi)

**Table of Contents**

[**System Layout** 3](#_Toc155271508)

[**Overview** 3](#_Toc155271509)

[**Components integration** 3](#_Toc155271510)

[ **Tiva C Microcontroller** 3](#_Toc155271511)

[ **Arduino Microcontroller** 10](#_Toc155271512)

[**List of Components** 13](#_Toc155271513)

[**Tiva C Components** 13](#_Toc155271514)

[**Arduino Components** 14](#_Toc155271515)

[**Circuits Wiring** 15](#_Toc155271516)

[ **Tiva C Components Wiring** 15](#_Toc155271517)

[**1.** **Infrared Transmitter** 15](#_Toc155271518)

[**2.** **HC-SR04 Ultrasonic Sensor** 15](#_Toc155271519)

[**3.** **KY-024 Linear Magnetic Hall Sensor** 16](#_Toc155271520)

[**4.** **MQ-2 Smoke Gas Sensor** 16](#_Toc155271521)

[**5.** **HC-05 Bluetooth Module** 17](#_Toc155271522)

[ **Arduino Components Wiring** 18](#_Toc155271523)

[**1.** **Infrared LED Diode 5mm** 18](#_Toc155271524)

[**2.** **LCD 16x2** 18](#_Toc155271525)

[**3.** **Buzzer** 19](#_Toc155271526)

[**4.** **LED + Button** 19](#_Toc155271527)

[**Mobile App Developed** 20](#_Toc155271528)

[**Main Flow of the Program** 21](#_Toc155271529)

[ **Initiating the program** 21](#_Toc155271530)

[ **Sensors Flow** 22](#_Toc155271531)

[**1.** **Ultrasonic Sensor** 22](#_Toc155271532)

[**2.** **Magnetic Sensor** 23](#_Toc155271533)

[**3.** **MQ-2 Smoke/Gas Sensor** 24](#_Toc155271534)

[**4.** **Bluetooth Module** 25](#_Toc155271535)

[ **Terminate the program** 25](#_Toc155271536)

[**Problem faced and how it was solved** 26](#_Toc155271537)

**List of Figures**

[Figure 1: Tiva Board 3](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612288)

[Figure 2: Ultrasonic Sensor Code 4](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612289)

[Figure 3: Magnetic Sensor Code 5](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612290)

[Figure 4: Smoke/Gas Sensor Code 6](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612291)

[Figure 5: Bluetooth Module Code 7](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612292)

[Figure 6: IR Transmitter Protocol Sample 8](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612293)

[Figure 7: Pushbuttons Code 9](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612294)

[Figure 8: Arduino Board 10](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612295)

[Figure 9: IR Receiver Code 11](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612296)

[Figure 10: LCD Initialization Code 11](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612297)

[Figure 11: Buzzer + Mute Button + LED Code 12](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612298)

[Figure 12: IR Transmitter 15](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612299)

[Figure 13: HC-SR04 Ultrasonic Sensor 15](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612300)

[Figure 14: KY-024 Magnetic Sensor 16](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612301)

[Figure 15: MQ-2 Smoke/Gas Sensor 16](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612302)

[Figure 16: HC-05 Bluetooth Module 17](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612303)

[Figure 17: Tiva Project 1 17](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612304)

[Figure 18: Tiva Project 2 17](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612305)

[Figure 19: 5mm IR Receiver 18](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612306)

[Figure 20: 16x2 LCD 18](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612307)

[Figure 21: Buzzer 19](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612308)

[Figure 22: LED 19](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612309)

[Figure 23: Pushbutton 19](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612310)

[Figure 24: Application Connection Message 20](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612311)

[Figure 25: Serial Bluetooth Terminal Menu 20](file:///C:\Users\HP\Desktop\Embedded%20Project%20Report.docx#_Toc154612312)

# **System Layout**

**Overview**  
In the Li-Fi project, the flow begins with the continuous monitoring of the environment by Ultrasonic, Magnetic, and Smoke sensors. When any of these sensors detect an event, a signal is generated, and the Tiva C microcontroller processes this information. Subsequently, the Tiva C prepares a data packet indicating which sensor triggered the event, and this packet is transmitted via an Infrared (IR) transmitter. On the receiving end, the Arduino, equipped with an IR receiver, captures and interprets the transmitted data. The Arduino then identifies the specific sensor responsible for the event and prints relevant information. Additionally, the project incorporates a Bluetooth module, facilitating communication between the Tiva C and a mobile device. The "Serial Bluetooth Terminal" application on the mobile device establishes a connection with the Tiva C, allowing users to monitor real-time sensor data remotely. This seamless flow from sensor detection to data transmission and mobile monitoring enhances the project's functionality and utility.

## **Components integration**

### **Tiva C Microcontroller**

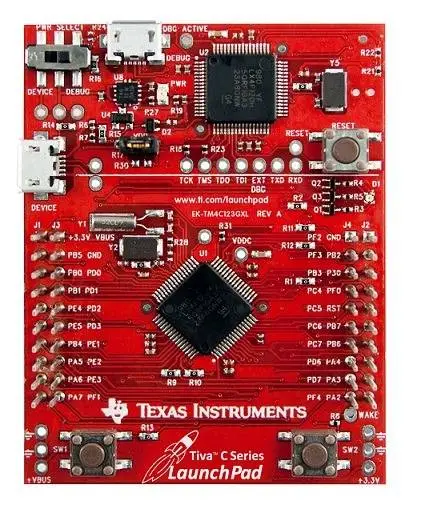
The Tiva C microcontroller interfaces with three sensors, a dedicated module, an infrared transmitter for sending data to the Arduino and 2 pushbuttons to start and stop the system.

Figure 1: Tiva Board

#### **Sensors**

##### **Ultrasonic**

Ultrasonic sensor functions by sending a 10-microsecond sound wave pulse through its trigger pin. It then waits for the return of the sound wave, detecting its reflection on the echo pin. This setup enables the Tiva C microcontroller to accurately detect objects. In the event of object detection, the Tiva C microcontroller generates binary pulses with a specific delay. Subsequently, these binary pulses are transmitted through the Infrared (IR) transmitter, allowing the Tiva C to communicate the detection event to the Arduino.

We utilized pin 4 within the GPIO register of Port A for the trigger pin. Consequently, pin 4 is configured as an output pin and for the echo pin, we assigned pin 6 in the GPIO register of Port B. As a result, pin 6 is configured to serve as an input pin

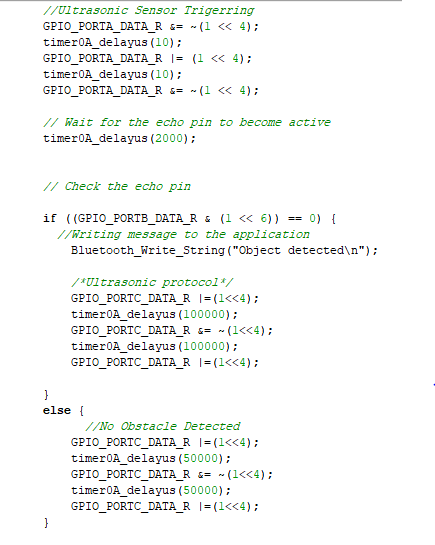


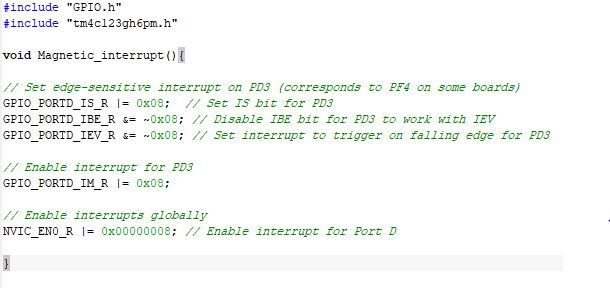
Figure 2: Ultrasonic Sensor Code

##### **Magnetic Sensor**

The Hall effect sensor, or magnetic sensor, utilizes a small semiconductor element. When a magnet approaches the sensor, it generates a magnetic field that influences the movement of charge carriers within the semiconductor. This deviation of charge carriers results in a measurable voltage across the semiconductor, indicating the presence of a magnetic field or the proximity of a magnet. The strength and polarity of the magnetic field influence the magnitude and direction of the voltage generated, allowing the sensor to detect both the presence and characteristics of a nearby magnet.

The voltage produced is then transferred to the Tiva C microcontroller as a digital or analog signal, depending on which pin is used. In our configuration, we use Port D, Pin 3, as an input in the GPIO register on the Tiva C microcontroller to receive and process the sensor output.

When the Tiva C receives input from the magnetic sensor, and it detects the presence of a magnet, it initiates specific delays for sending the data through the transmitter to the Arduino.



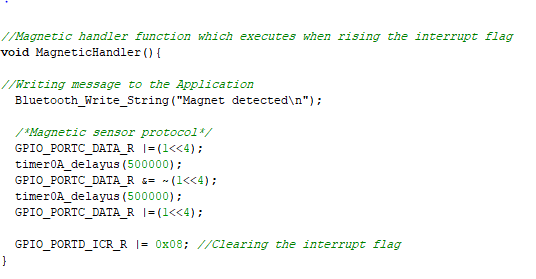


Figure 3: Magnetic Sensor Code

##### **Smoke/Gas Sensor**

The MQ-2 gas sensor, relying on semiconductor technology with a tin dioxide (SnO2) sensing element, features a built-in heating element that elevates the semiconductor's temperature. Upon exposure to target gases such as smoke or combustible gases, interactions with the tin dioxide surface induce changes in conductivity. The resultant alteration is transformed into an electrical signal, offering insight into the concentration of the target gas. Calibration is crucial to establish a precise correlation between the sensor's output signal and the actual gas concentration. Additionally, when smoke is detected, the analog-to-digital value in the sequencer experiences fluctuations, increasing when smoke is present and decreasing in its absence. Based on the analog value, data is transmitted to the Arduino through the IR transmitter with certain binary pulses, contributing to a comprehensive gas detection and monitoring system.

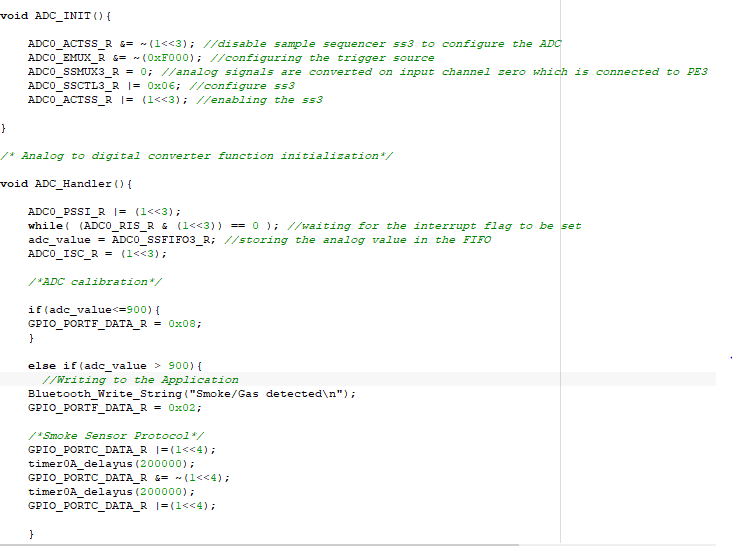
We used AN0 input channel in ADC0 module which is connected to Pin 3 in Port E GPIO.

Figure 4: Smoke/Gas Sensor Code

#### **Bluetooth Module**

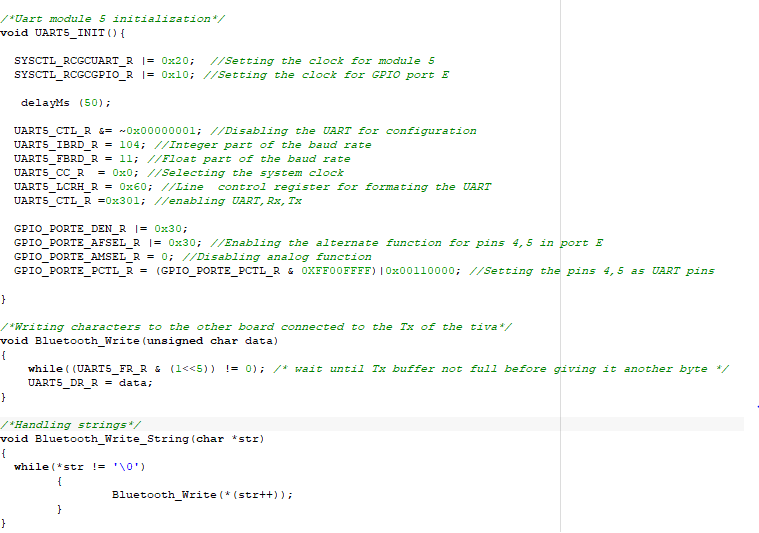
  
The Bluetooth module employed in the system operates in full-duplex mode, allowing bidirectional communication—both sending and receiving data. This bidirectional communication is facilitated through the utilization of UART (Universal Asynchronous Receiver-Transmitter) registers, enabling seamless interaction with the microcontroller. Consequently, the microcontroller can transmit data to the mobile application through the Bluetooth module. The designated application for this purpose is called Serial Bluetooth Terminal, available on the App Store for convenient accessibility.

Figure 5: Bluetooth Module Code

#### **Infrared Transmitter**

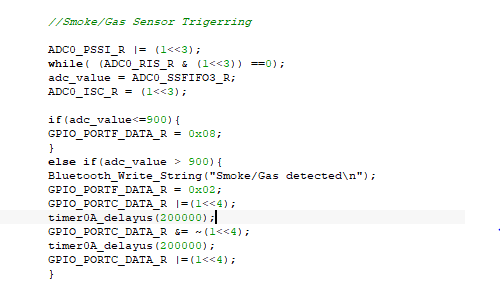
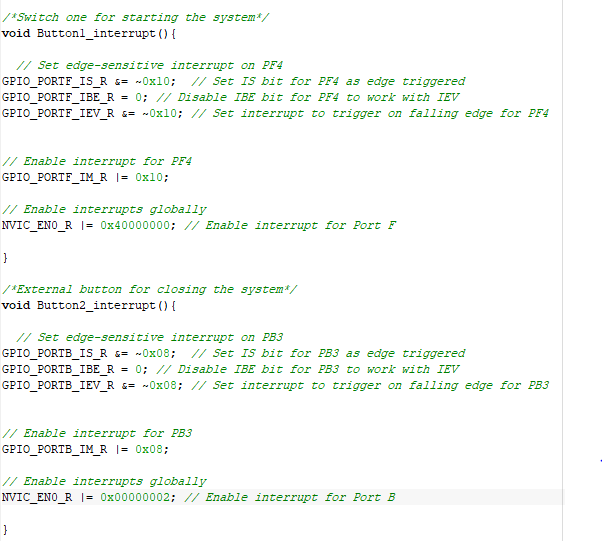
  
The infrared transmitter acts as a conduit for receiving pulses from the Tiva C, with the pulse pattern contingent on the protocol established for each sensor during value readings. Subsequently, these pulses are transmitted to the receiving end on the Arduino. The specific pulse sequence is determined by the millisecond delay assigned to the transmitter, a parameter influenced by the characteristics of the connected sensors. This synchronization ensures accurate data transmission from the Tiva C to the Arduino, facilitating seamless communication between the microcontrollers and enhancing the overall functionality of the system

Figure 6: IR Transmitter Protocol Sample

#### **Pushbuttons**

We are using already implemented switch 1 on the tiva c for initiating the system, and for stopping the system we use an external pushbutton connected to pin 3 on GPIO port B. the pushbuttons are configured as pullup resistance, therefore, they are high when no pressing and low when pressed so we set the interrupt on the falling edge of the pushbuttons



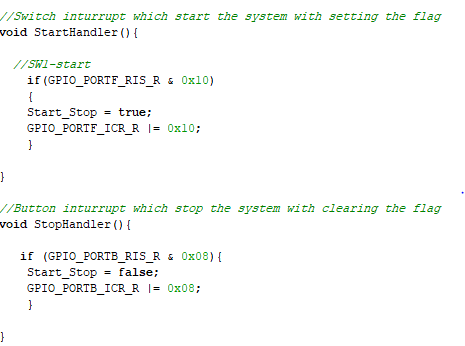


Figure 7: Pushbuttons Code

## **Arduino Microcontroller**

On the Arduino, we have integrated an LCD to showcase the data received from the Tiva through the transmitter via the IR receiver. Additionally, we incorporated a buzzer that activates whenever a sensor transmits data from the Tiva, accompanied by an LED that illuminates in sync with the sensor data. To enhance user control, a pushbutton is implemented to mute the buzzer, introducing a 5-second delay for effective interaction. Lastly, the Arduino is equipped with an IR receiver to seamlessly capture the data transmitted by the Tiva through the IR transmitter, completing the integrated functionalities of our system



Figure 8: Arduino Board

##### **Infrared Receiver**

The infrared receiver on the Arduino captures pulses transmitted from the infrared transmitter integrated into the Tiva. These pulses are subsequently interpreted using the pulseIn() function in the Arduino, allowing measurement of the duration of both high and low pulses based on the specific configuration chosen for the system, and the resulting pulses are stored in an unsigned long variable. This choice is made to prevent potential overloads, as using an int variable may lead to negative numbers due to the inherent characteristics of pulse duration values.

We used pin 9 as a digital input pin for the IR receiver

Figure 9: IR Receiver Code

##### **LCD**

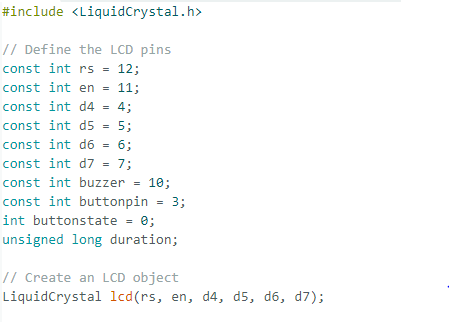
To facilitate the operation of the LCD on the Arduino, it is essential to include the LiquidCrystal library, which provides necessary functions for controlling the display. In our implementation, we utilized only four pins from the D0-D7 pins, specifically D4-D7, to interface with the LCD. Additionally, a potentiometer is integrated with the LCD setup to regulate the brightness of the display, offering flexibility in adjusting visual parameters to suit varying environmental conditions

Figure 10: LCD Initialization Code

##### **Buzzer + Mute Button + LED**

We integrated a buzzer and LED into our system, connected to pin 10 on the Arduino and configured as an OUTPUT. This buzzer and the LED are designed to become active (high) whenever any sensor sends values. To enhance user control, we added a pushbutton on pin 3, directly linked to the buzzer, allowing users to mute the buzzer by pressing the button. Notably, the muting action takes effect after a 5-second delay from the moment the pushbutton is pressed.

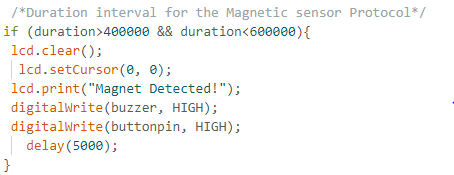


Figure 11: Buzzer + Mute Button + LED Code

# **List of Components**

|  |  |  |
| --- | --- | --- |
| **Tiva C Components** | | |
| Quantity | Name | Picture |
| 1 | 120Ω Resistance |  |
| 1 | Infrared Transmitter |  |
| 1 | HC-SR04 Ultrasonic Sensor |  |
| 1 | KY-024 Linear Magnetic Hall Sensor |  |
| 1 | MQ-2 Smoke Gas Sensor |  |
| 1 | HC-05 Bluetooth Module |  |
| 1 | Breadboard |  |

|  |  |  |
| --- | --- | --- |
| **Arduino Components** | | |
| Quantity | Name | Picture |
| 1 | 10KΩ Resistance,120Ω, and 2.2KΩ |  |
| 1 | Infrared LED Diode 5mm |  |
| 1 | LCD 16x2 |  |
| 1 | Buzzer |  |
| 1 | Push Button |  |
| 1 | LED |  |
| 1 | Breadboard |  |
| 1 | Potentiometer |  |

# **Circuits Wiring**

## **Tiva C Components Wiring**

Initially, we will establish the power supply connection by connecting the 3.3V output on the Tiva C to the positive (+) rail on the breadboard. Simultaneously, the ground (GND) of the Tiva C will be connected to the negative (-) rail of the breadboard. This configuration allows us to conveniently place and power the various modules on the breadboard.

### **Infrared Transmitter**

Figure 12: IR Transmitter

The positive (+) terminal of the infrared transmitter's long LED is connected to a 120-ohm resistor. The other end of this resistor is then connected to pin 4 in GPIO Port C, establishing a wiring link for effective circuit integration. Simultaneously, the short leg (-) of the LED is connected to the ground, completing the circuit connection for the infrared transmitter.

### **HC-SR04 Ultrasonic Sensor**

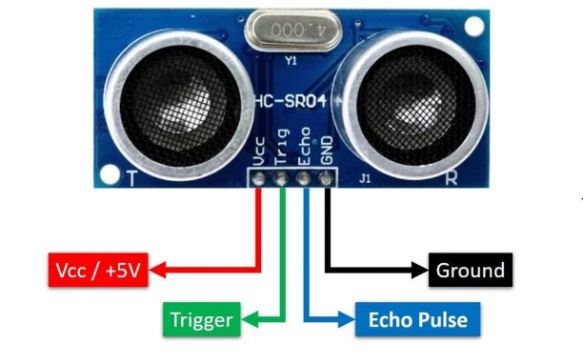
As illustrated in the corresponding schematic, the left and right pins of the ultrasonic sensor are designated as VCC and GND, respectively. Consequently, we connect the VCC pin to the positive (+) rail of the breadboard and the GND pin to the negative (-) rail. Moving on to the other two pins, the trigger pin serves as an output, responsible for emitting the pulse generated. Hence, we connect the trigger pin of the ultrasonic sensor to an output pin on the Tiva C, which is in our case pin 4 in GPIO Port A. This pin is configured in the code as an output. As for the echo pin, it functions as an input, receiving the sound wave reflected from the object. We connect it to an input pin on the Tiva C, which is in our case pin 6 in GPIO Port B, which has been configured also as an input in the code. It is crucial to note that the echo pin sends a 5V signal to the Tiva C. Therefore, we must exercise caution and avoid using pins PD4, PD5, PB0, and PB1, as these are limited to a maximum of 3.6

Figure 13: HC-SR04 Ultrasonic Sensor

### **KY-024 Linear Magnetic Hall Sensor**

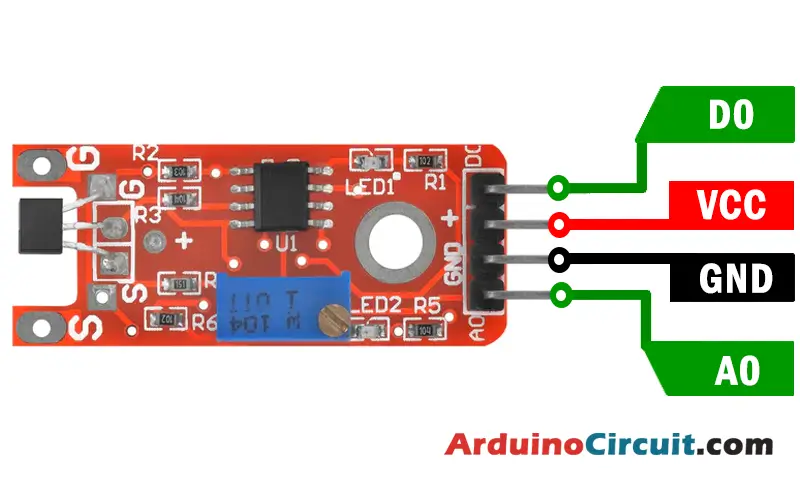
For the magnetic sensor, we establish a connection by linking its VCC to the positive (+) rail on the breadboard and GND to the negative (-) rail. Subsequently, two additional pins, A0 and D0, are available for analog and digital use, respectively. In our configuration, we opt for digital use and connect the D0 pin to the Tiva C. This sensor interprets the magnetic field as an input and generates a voltage output for the Tiva. Consequently, the D0 pin is designated as an input pin, specifically a digital GPIO pin. To establish this connection, we connect the D0 pin on the sensor to pin 3 on GPIO Port D, and the necessary pin configuration is implemented in the code

Figure 14: KY-024 Magnetic Sensor

### **MQ-2 Smoke Gas Sensor**

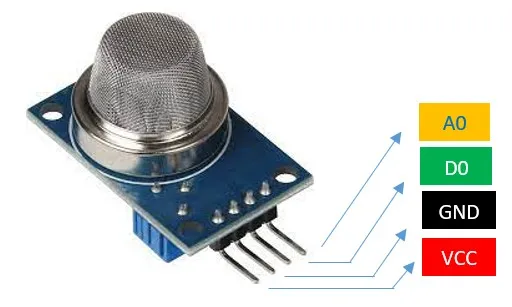
For the MQ-2 smoke sensor, our wiring configuration involves connecting its VCC to the positive (+) rail on the breadboard and GND to the negative (-) rail. The sensor provides two pins, A0 for analog use and D0 for digital use. In this setup, we opt for analog connection. Utilizing the ADC input channels, specifically AN0, which corresponds to pin 3 in Port E on the Tiva C, we establish the necessary connection. It's important to note that AN0 is configured as part of ADC module 0, not as a general-purpose input/output (GPIO) since it doesn't serve as a digital input or output. Given that AN0 connects to pin 3 in Port E, we connect the A0 pin of the sensor to pin 3 in Port E on the Tiva C, ensuring the proper integration of the MQ-2 smoke sensor with the microcontroller.

Figure 15: MQ-2 Smoke/Gas Sensor

### **HC-05 Bluetooth Module**

Figure 16: HC-05 Bluetooth Module

In the HC-05 Bluetooth module, we find a total of 6 pins, with only 4 being essential for our setup. Connecting the VCC and GND pins to the positive (+) and negative (-) rails on the breadboard, respectively, provides the necessary power supply. The module features Tx (transmitter) and Rx (receiver) pins, crucial for communication. To integrate the Bluetooth module with the Tiva C, we employ UART (Universal Asynchronous Receiver-Transmitter) modules. In our specific configuration using UART module 5, we connect the Tx pin on the Tiva C (pin 5, Port E) to the Rx pin on the Bluetooth module. Similarly, the Rx pin on the Tiva C (pin 4, Port E) is connected to the Tx pin on the Bluetooth module. It is imperative to note that the VCC pin on the Bluetooth module operates on 5V. Additionally, the specific UART pins on the Tiva C, as per the datasheet, should be verified before usage to ensure correct interfacing

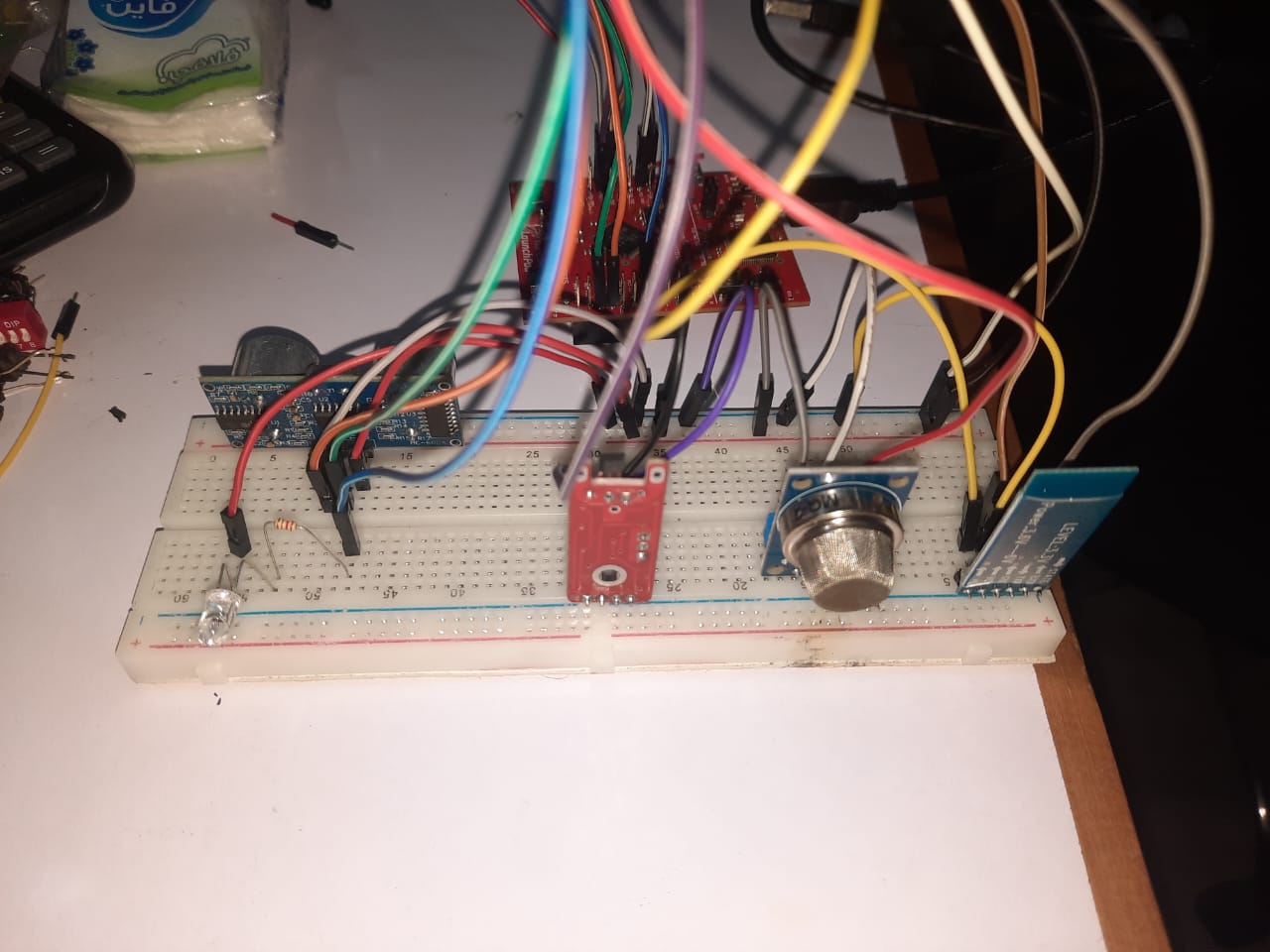


Figure 17: Tiva Project 1

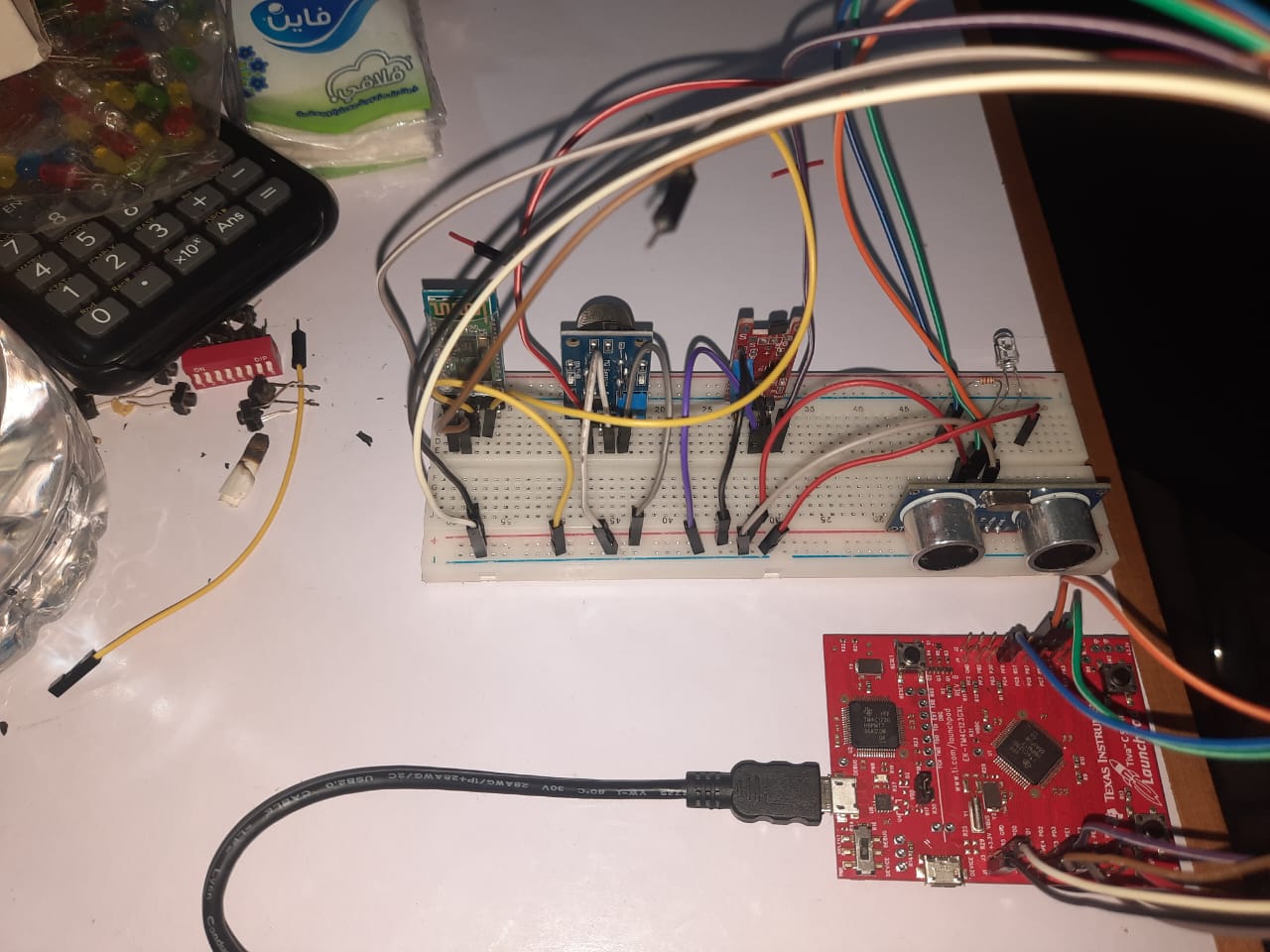


Figure 18: Tiva Project 2

## **Arduino Components Wiring**

Initially, we will establish the power supply connection by connecting the 5V output on the Arduino to the positive (+) rail on the breadboard. Simultaneously, the ground (GND) of the Arduino will be connected to the negative (-) rail of the breadboard. This configuration allows us to conveniently place and power the various modules on the breadboard.



Figure 19: 5mm IR Receiver

### **Infrared LED Diode 5mm**

For the IR LED receiver, the wiring configuration involves connecting the short leg (-) to the positive (+) rail of the breadboard. The long leg (+) is connected differently: a 10k-ohm resistor is placed in series with the input pin used to receive values, in our case it is pin 9 on the Arduino board. The other end of this resistor is then connected to the negative (-) rail of the breadboard. This arrangement ensures the proper reception of values through pin 9 while incorporating the necessary resistance to manage the signal flow effectively.

### **LCD 16x2**

In setting up the LCD, we begin by connecting the VCC (VDD) to the positive (+) rail of the breadboard and the GND (VSS) to the negative (-) rail. The contrast pin (V0) is linked to the middle pin of a potentiometer, serving to adjust the brightness. The LCD relies on the potentiometer as it acts as a resistance for the reading process. We specifically utilize the RS (Register Select) pin, connecting it to pin 12 on the Arduino, as it is crucial for our operation. The Enable (EN) pin is essential for enabling the Arduino, and therefore, we connect it to pin 11. As for the data pins (D0-D7), our setup only requires D4-D7, and these are connected in order from pins 4-7 on the Arduino, establishing a comprehensive and functional connection for the LCD, and the last 2 pins are anode, and cathode, so we connect Anode to (+) and cathode to (-).

Figure 20: 16x2 LCD

### **Buzzer**

Figure 21: Buzzer

The setup for the buzzer is straightforward. With two legs, the long leg is connected to pin 10 on the Arduino board, while the short leg is connected to the negative (-) rail on the breadboard

### **LED + Button**

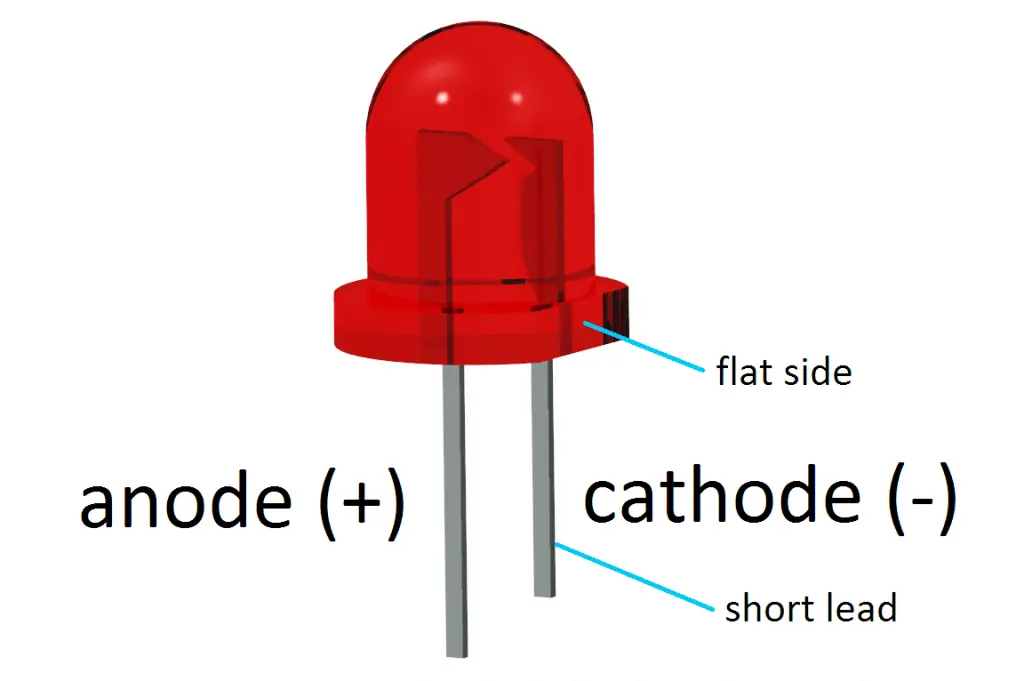
  
The connection for the LED involves attaching the long leg to a 120-ohm resistor, with the other end of the resistor connecting to pin 8 on the Arduino. Simultaneously, the short leg is linked to the negative (-) rail on the breadboard. For the pushbutton, which consists of two identical legs, one leg is connected to the positive (+) rail on the breadboard. The other leg is connected in to a 2.2k-ohm resistor which is connected in series linked to pin 3 on the Arduino.

Figure 22: LED

Figure 23: Pushbutton

# **Mobile App Developed**

The mobile app utilized for this system is called 'Serial Bluetooth Monitor,' available for installation on Android phones through the Play Store. To initiate communication, ensure that your Bluetooth is enabled on your mobile device and proceed to pair it with the HC-05 module. The default password for pairing is either '0000' or '1234

After successfully pairing, open the application and navigate to the top-left menu. Select 'Devices' from the menu, and you will find the list of already paired devices. Connect to the HC-05 from the list, and a message on the terminal will confirm 'Connected.' Now, your system is ready for operation.

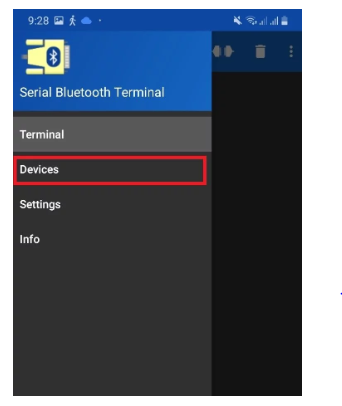
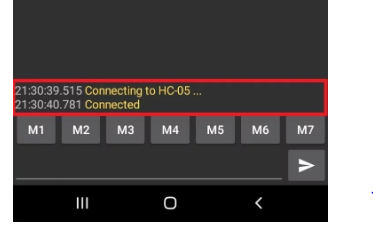


Figure 24: Application Connection Message

Figure 25: Serial Bluetooth Terminal Menu

# **Main Flow of the Program**

## **Initiating the program**

First, let's discuss the system initiation process, which involves pressing switch one on the tiva board to start the system.

1. Connecting the Board to the laptop
2. Build and run the code
3. Press SW1 on the tiva board
4. Interrupt flag pin becomes high
5. Handler function corresponding to the interrupt flag is called from the NVIC vector table
6. Handler function executes
7. Start\_Stop flag becomes true
8. Interrupt flag is cleared
9. System starts
10. Uart module is initialized
11. GPIO’s are initialized
12. ADC module is initialized
13. Interrupts are enabled

## **Sensors Flow**

### **Ultrasonic Sensor**

1. Giving the trigger pin which is connected to pin 4 GPIO port A, 10 microseconds pulse
2. Trigger pin produces sound wave for detection
3. Echo pin which is connected to pin 6 GPIO port B is at high state
4. Delay for 2 milliseconds
5. When the sound wave is received on the echo pin due to object detection pin 6 in port B becomes zero
6. If pin 6 is set to zero is a certain protocol using pulses is made and sent through the infrared transmitter to the infrared receiver to indicate object detection
7. Infrared receiver receives the pulses on the Arduino side
8. On the received pulses the LCD prints “Object detected”
9. Buzzer is at high state
10. LED flashes
11. If mute button is pressed the buzzer and the LED are turned off after 5 seconds
12. If pin 6 in GPIO port B is equal to 1
13. A certain protocol is sent through IR transmitter
14. IR receiver receives the pulses
15. LCD prints “Nothing detected”

### **Magnetic Sensor**

1. Connecting pin D0 on the magnetic sensor with pin 3 GPIO port D
2. Configuring pin 3 as digital enable input pin
3. If a magnet is detected pin 3 will have falling edge
4. Interrupt flag is set
5. Handler function corresponding to Port D in the NVIC vector table is called
6. MagneticHandler function sends a certain protocol using pulses is sent through the IR transmitter to the IR receiver
7. Interrupt flag is cleared
8. Infrared receiver receives the pulses on the Arduino side
9. On the received pulses the LCD prints “Magnet detected”
10. Buzzer is at high state
11. LED flashes
12. If mute button is pressed the buzzer and the LED are turned off after 5 seconds
13. If there was no interrupt a certain protocol is sent through IR transmitter
14. IR receiver receives the pulses
15. LCD prints “Nothing detected”

### **MQ-2 Smoke/Gas Sensor**

1. Connecting pin A0 on the Smoke sensor with pin 3 GPIO port E
2. Configuring pin 3 as ADC input channel ss3 pin
3. Sensor reads analog values by default
4. When this adc values exceeds 900 then there is a smoke to detect
5. A certain protocol using pulses is sent through the IR transmitter to the IR receiver
6. Infrared receiver receives the pulses on the Arduino side
7. On the received pulses the LCD prints “Smoke/Gas detected”
8. Buzzer is at high state
9. LED flashes
10. If mute button is pressed the buzzer and the LED are turned off after 5 seconds
11. If the adc value was below 900 a certain protocol is sent through IR transmitter
12. IR receiver receives the pulses
13. LCD prints “Nothing detecte

### **Bluetooth Module**

1. Initialize UART Module 5 for communication.
2. Configure UART pins for Module 5. Set GPIO E pin 4 as Rx (Receive) and pin 5 as Tx (Transmit).
3. Physically connect the Rx (pin 4) on the Tiva with the Tx pin on the Bluetooth module, and Tx (pin 5) on the Tiva with the Rx pin on the Bluetooth module.
4. Wait until the FIFO (First In, First Out) receive register is not full, indicating that it can receive bytes.
5. Send the received bytes to the Bluetooth module.
6. The message is received on the application connected to the HC-05 Bluetooth module.

## **Terminate the program**

1. Connect a pushbutton to pin 3 GPIO port B
2. Configure the pin as digital enable input pin with pullup resistor
3. Interrupt flag pin becomes high
4. Handler function corresponding to the interrupt flag is called from the NVIC vector table
5. Handler function executes
6. Start\_Stop flag becomes false
7. Interrupt flag is cleared
8. System terminates

# **Problem faced and how it was solved**

In the initial stages of the project, we encountered challenges in determining which registers to use and how to configure them, leading to uncertainty about the correct values. This was addressed by delving into the datasheets, gaining a deeper understanding of each register's purpose and making informed decisions. Another issue arose when the IR receiver suddenly ceased functioning. Initially suspecting a hardware malfunction, extensive testing revealed that the problem lay in the register configuration. Adjusting the register to a 10Kohm setup resolved the issue, demonstrating the importance of meticulous configuration. In the case of the ultrasonic sensor, an initial hurdle was the absence of sound waves being transmitted. The solution involved ensuring that the trigger pin received a 10-microsecond pulse, followed by an adequate delay to allow the echo pin to receive the waves effectively. Additionally, dealing with unexpected output required debugging at the register level, a crucial step in uncovering the exact values being written and their implications. This challenging yet valuable experience taught us to approach debugging not only in the code but also by examining and understanding the register-level interactions, enhancing our overall learning from the project.