**VISVESVARAYA TECHNOLOGICAL UNIVERSITY BELAGAVI, KARNATAKA– 590018**



**Report on**

**“IOT Practical Assessment”**

**Submitted By**

**NAME[USN]**

***Submitted in partial fulfillment of the requirement for the award of degree of***

**BACHELOR OF ENGINEERING COMPUTER SCIENCE AND ENGINEERING**

**UNDER THE GUIDANCE**

**Mrs. Shruthi H S**

Assistant Professor, Dept. of CSE

Rajeev Institute of Technology, Hassan



**RAJEEV INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**HASSAN-573201 2022-2023**

**RAJEEV INSTITUTE OF TECHNOLOGY,**

**HASSAN**

**(Approved by AICTE, New Delhi and Affiliated to VTU, Belagavi.)**

**Plot #1-D, Growth Centre, Industrial Area, B-M Bypass Road, Hassan – 573201**

**Ph: (08172)-243180/83/84 Fax:(08172)-243183**

Icon

Description automatically generated with medium confidence

**Department of Computer Science and Engineering**

CERTIFICATE

Certified that the Report titled **“IOT PRACTICAL ASSESSMENT”** is carried out by. **NAME, USN: \_\_\_\_\_\_\_**, a Bonafede student of **RAJEEV INSTITUTE OF TECHNOLOGY**, Hassan in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering of Visveswaraya Technological University, Belagavi** during the year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The report has been approved as it satisfies the academic requirements in respect of Assessment prescribed for the said Degree.

**Mrs. SHRUTHI H S Dr. H N PRAKASH Dr. MAHESH P K**

Assistant Professor Head of the Department Principal

Dept. of Computer Science & Computer Science & Engineering **RIT, HASSAN**

Engineering **RIT, HASSAN**

**RIT, HASSAN**

**Examiners Signature with Date**

**1.**

**2.**

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

We would like to profoundly thank our **Management of Rajeev Institute of Technology** & our President **Dr. Rachana Rajeev** for providing such a healthy environment.

We would like to express our sincere thanks to our principal **Dr. Mahesh P K**, Rajeev Institute of Technology for his encouragement that motivated us for successful completion of project work.

We wish to express our gratitude to **Dr. H N Prakash**, Head of the Department of Computer Science & Engineering for providing a good working environment and for his constant support and encouragement.

It gives us great pleasure to express our gratitude to **Mrs. Shruthi H S** Assistant Professor, Department of Computer Science & Engineering for her expert guidance, initiative and encouragement that led us to complete this project.

We would also like to thank all our staffs of Information Science and Engineering department who have directly or indirectly helped us in the successful completion of this project and also, we would like to thank our parents.

NAME

USN

# INDEX

|  |  |  |
| --- | --- | --- |
| **Sl No.** | **Index** | **Page No** |
| 1 | Objective | 1 |
| 2 | Aim | 1 |
| 3 | Transmit a string using UART | 2 |
| 4 | Point-To-Point Communication of Two Motes over the Radio  Frequency | 4 |
| 5 | Multi Point to Single point communication of Motes over the Radio  Frequency: | 7 |
| 6 | Communication of Motes over the Radio Frequency I2C Protocol | 10 |
| 7 | Reading Temperature and Relative Humidity Value from the sensor | 14 |
| 8 | Outcome | 17 |
| 9 | Conclusion | 18 |
|  | Reference |  |

**Objectives:**

# PRACTICAL ASSESSMENTS

The main objective of completing these programming projects is to gain practical experience in working with different communication protocols and sensors commonly used in embedded systems development. By completing these exercises, one can develop valuable skills in programming for embedded systems that can be useful for building various devices and projects, such as home automation systems, smart sensors, and IoT devices.

Furthermore, the objective is to learn how to establish communication between different devices using different communication protocols, such as UART, I2C, and radio frequency. This includes understanding how to transmit data between devices, establish point-to-point or multi-point communication, and interface with various sensors to extract data from them.

Overall, the objective is to gain a strong foundation in programming for embedded systems and develop skills that are highly sought after by employers in the technology industry. Through completing these programming projects, one can develop a practical understanding of various communication protocols and sensors, and gain experience in working with them to build real-world applications

## Aim:

The aim of doing practical assessments in IoT (Internet of Things) is to evaluate the practical skills and knowledge of the individuals or students in implementing IoT-based solutions. Practical assessments in IoT typically involve building and testing hardware devices and software applications that can communicate with each other using various protocols and interfaces.

Overall, practical assessments in IoT provide a valuable opportunity for individuals or students to apply their theoretical knowledge and skills to real-world scenarios, and to gain practical experience in a rapidly growing and evolving field.

# PROGRAM 1

## Transmit a string using UART: Components Required:

Arduino UNO

Arduino UNO USB Cable

## Procedure:

* 1. Open the Arduino UNO IDE and Create and save a sketch source code as follows.
  2. compile sketch
  3. upload sketch on to Arduino UNO

## Program:

import serial

# Initialize UART serial port with baud rate and timeout ser = serial. Serial ('/dev/ttyUSB0', 9600, timeout=1)

# Define the string to transmit tx\_string = "Hello, world!"

# Convert the string to bytes tx\_bytes = tx\_string. encode('utf-8') # Transmit the bytes over UART ser. writes(tx\_bytes)

# Close the serial port ser. close ()

## Explanation:

In this code, we use the serial module in Python to communicate over UART. We first initialize the serial port with the appropriate settings, such as the baud rate and timeout. Then, we define the string we want to transmit and convert it to bytes using the encode () method. Finally, we use the write () method of the serial port object to send the bytes over UART. Don't forget to close the serial port at the end of the program using the close () method.

## Output:

To receive the transmitted string over UART, you'll need a device on the other end of the serial connection to receive and display the data. Here's an example Python code to receive and display the string transmitted by the previous code:

import serial

# Initialize UART serial port with baud rate and timeout ser = serial. Serial ('/dev/ttyUSB0', 9600, timeout=1)

# Read the incoming bytes from UART rx\_bytes = ser. read (14)

# Convert the bytes to string rx\_string = rx\_bytes. decode('utf-8') # Print the received string

print ("Received string: ", rx\_string) # Close the serial port

ser. close ()

In this code, we initialize the serial port with the same settings as in the previous code. Then, we use the read () method to read 14 bytes from the serial port, which should correspond to the length of the transmitted string. We convert the received bytes back to a string using the decode () method, and print the received string. Finally, we close the serial port using the close () method.

COMS (Arduino/Genuino Uno) Enter your Bane:

Enter your Height Feet. Inch:

Hello Akhil

21 years old and You are 5.6 tall

## Applications:

UART’s are used for devices including GPS units, modems, wireless communication and Bluetooth Modules, amongst many other applications.

## Conclusion:

Serial data communication is achieved by transmitting and receiving String data using UART.

# PROGRAM 2

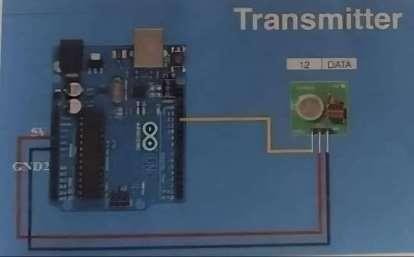
## Point-To-Point Communication of Two Motes over the Radio Frequency. Components Required:

Arduino UNO’s

RF Transmitter 433MHz RF receiver 433MHz Connecting wires

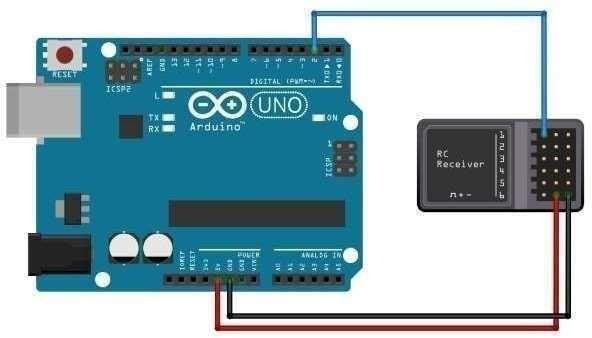
## Arduino Sender Connections:

|  |  |
| --- | --- |
| Arduino Pins | RF Transmitter |
| 12 | Data |
| 5V | VCC |
| GND2 | GND |



**Arduino Receiver Connection:**

|  |  |
| --- | --- |
| Arduino Piers | RF Receiver |
| 11 | Data |
| GND2 | GND |



Receiver

## Procedure:

* 1. Open the Arduino UNO IDE and create and save sketch source code of transmitter and receiver.
  2. Compile the sketches
  3. Upload sketches onto Arduino UNO’s through connected COM parts.

## Program:

To establish point-to-point communication between two motes over radio frequency, you can use a wireless communication protocol such as Zigbee or Bluetooth Low Energy (BLE). Here's an example code for establishing a simple communication link between two motes using the Zigbee protocol:

First Mote:

from xbee import ZigBee import serial

# Initialize the serial port for the XBee radio ser = serial. Serial ('/dev/ttyUSB0', 9600)

# Create a ZigBee object for the radio xbee = ZigBee(ser)

# Define the destination address for the other mote dest\_addr = b'\x00\x01'

# Send a message to the other mote

xbee. send ('tx', dest\_addr=dest\_addr, data='Hello, world!') # Wait for a response from the other mote

response = xbee. wait\_read\_frame () # Print the response

message print(response['rf\_data']) # Close the serial port

ser. close ()

## Explanation:

In this code, we use the xbee module in Python to communicate over Zigbee. We first initialize the serial port for the XBee radio with the appropriate settings, such as the baud rate. Then, we create a ZigBee object for the radio. We define the destination address for the other mote, which should be set to its unique 64-bit address. We use the send () method

of the ZigBee object to send a message to the other mote, specifying the destination address and the data to be sent. We then wait for a response from the other mote using the wait\_read\_frame () method, which blocks until a message is received. Finally, we print the received message and close the serial port.

## Output:

The output of the program will depend on the response received from the other mote. If the other mote is also running a similar program and responds with a message, then the output of the program will be the received message printed to the console.

For example, if the other mote responds with the message "Hi there!", then the output of the program would be:

**Hi there!**

## Applications:

RF technology, the information can be transmitted through the air without requiring any cable or wires or other electronic conductors, by using electromagnetic waves like IR, RF, satellite, etc. a variety of wireless communication devices and technologies ranging from smart phones to computers, tabs, laptops, Bluetooth Technology, printers.

## Conclusion:

Wireless communication is achieved by RF transmitter and RF receiver message passed in the frequency range of 433MHZ.

# PROGRAM 3

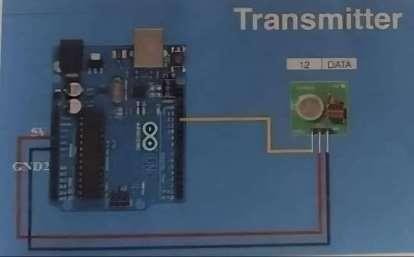
## Multi Point to Single point communication of Motes over the Radio Frequency:

**Components Required:**

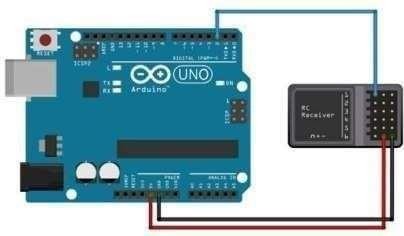
|  |  |  |
| --- | --- | --- |
| Sl. No | Components Required | QTY |
| 1 | Arduino UNO transmitter | 1 |
| 2 | Arduino UNO Receiver 1 | 1 |
| 3 | Arduino UNO Receiver 2 | 1 |
| 4 | Arduino UNO USB Cables | 3 |
| 5 | RF Receiver 433MHz | 2 |
| 6 | RF Transmitter 433MHz | 1 |
| 7 | Connecting Wires | 9 |

## Arduino Connectors:

|  |  |
| --- | --- |
| Arduino Pins | RF Transmitter |
| 12 | DATA |
| 5V | VCC |
| GND2 | GND |



|  |  |
| --- | --- |
| Arduino Pins | RF Receiver |
| 11 | Data |
| 5V | VCC |
| GND2 | CND |



**Procedure:**

Receiver

Open the Arduino UNO IDE and create and save a sketch of master transmitter and receiver 1 and receiver 2 source code.

Comp the sketch of transmitter and receivers.

Upload the sketch through COM ports (COM4, COM3, COM5)

See the output message onto the serial monitor by using COM ports.

## Program:

To establish multi-point to single-point communication between multiple motes and a single mote over radio frequency, you can use a wireless communication protocol such as Zigbee or Bluetooth Low Energy (BLE). Here's an example code for establishing a simple communication link between multiple motes and a single mote using the Zigbee protocol:

Single Mote:

from xbee import ZigBee import serial

# Initialize the serial port for the XBee radio ser = serial. Serial ('/dev/ttyUSB0', 9600)

# Create a ZigBee object for the radio xbee = ZigBee(ser)

# Set the mote to receive messages from any source addressxbee.at (command='DH', parameter=b'\x00\x00') xbee.at (command='DL', parameter=b'\x00\x00') xbee.at (command='MY', parameter=b'\x01\x02')

# Continuously receive and print messages from any source address while True:

try:

response = xbee. wait\_read\_frame () print(response['rf\_data']) except KeyboardInterrupt:

break

# Close the serial port ser. close ()

## Explanation:

In this code, we use the xbee module in Python to communicate over Zigbee. We first initialize the serial port for the XBee radio with the appropriate settings, such as the baud rate. Then, we create a ZigBee object for the radio. We use the at () method of the ZigBee object to set the destination address for the mote to its own address (01 02), and to set the source address to any address (00 00). This will enable the mote to receive messages from any source address. We use a loop to continuously receive and print messages from any source address using the wait\_read\_frame () method. Finally, we close the serial port.

## Output:

The output of the program will depend on the messages sent and received by the motes over the Zigbee protocol.

For the single mote code, the output will be any messages received by the mote from other motes over the Zigbee protocol. If there are no messages being sent by other motes, then the program will simply wait indefinitely for a message.

For the multiple motes code, the output will depend on whether the message was successfully sent to the single mote or not. If the message was successfully sent, then there will be no output. If there was an error in sending the message, then an exception will be raised and an error message will be printed to the console.

## Applications:

PMP Wireless networks are employed in distribution amenities, huge corporate campuses, school districts, public safety applications etc.

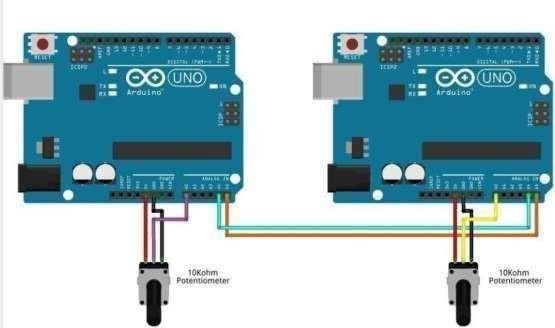
## Conclusion:

Wireless Network Communication is achieved through multi point to single point communication nodes over radio frequency by RF transmitter and receiver. Multiple receivers and a transmitter communicate with each other by broadcasting message in a network via RF transmitter and receiver notes in the frequency of 433MHz range.

# PROGRAM 4

## Communication of Motes over the Radio Frequency I2C Protocol Components Required:

|  |  |  |
| --- | --- | --- |
| Sl No | Components Required | QTY |
| 1 | Arduino UNO Master and PORT | 1 |
| 2 | Arduino UNO Slave Sender and PORT(COM3) | 1 |
| 3 | Arduino UNO USB Cable | 3 |
| 4 | LED Lights | 2 |
| 5 | Bread Board | 1 |
| 6 | Connecting Wires | 9 |



**Connections:**

|  |  |
| --- | --- |
| Arduino Master Pins | Arduino Slave Pins |
| A4 | A4 |
| A5 | AS |
| GND | GND |
| Arduino Master Pins | LED |
| 13 | +VE |
| GND | -VE |

## Procedure:

Set up the connection as per diagram through bread board and connect LEDs to Master and Slave Reader.

Compile and upload sketch Arduino UNO Master Sender/Reader through COM4. Compile and upload sketch Arduino UNO Slave Sender through COM3.

Compile and upload sketch Arduino UNO Slave Reader through COMS. See the Output status message on Serial Monitor and LED light status ON. (LED ONS) of Master sender and Slave sender UNO boards.

## Program:

Here's an example program in Python for using the I2C protocol to communicate betweena master and a slave device:

Import smbus import time

# Define the I2C bus number and device addresses BUS\_NUM = 1

MASTER\_ADDR = 0x50 SLAVE\_ADDR = 0x60

# Initialize the I2C bus

bus = smbus. SMBus (BUS\_NUM)

# Define a function to write a byte to the slave device def write\_byte(data):

bus. write\_byte (SLAVE\_ADDR, data) time. sleep (0.01)

# Define a function to read a byte from the slave devicedef read\_byte ():

return bus. read\_byte (SLAVE\_ADDR)

# Define a function to write a sequence of bytes to the slave device def write\_bytes(data):

bus. write\_i2c\_block\_data (SLAVE\_ADDR, 0,data) time. sleep (0.01)

# Define a function to read a sequence of bytes from the slave device def read\_bytes(num\_bytes):

return bus. read\_i2c\_block\_data (SLAVE\_ADDR, 0, num\_bytes) # Write a byte to the master device

bus. write\_byte (MASTER\_ADDR, 0xAB) # Read a byte from the master device

data = bus. read\_byte (MASTER\_ADDR)

print ("Received data: 0x{:02x}”. format(data)) # Write a sequence of bytes to the slave device write\_bytes ([0x12, 0x34, 0x56, 0x78])

# Read a sequence of bytes from the slave device

data = read\_bytes (4)

print ("Received data:", data)

## Explanation:

In this code, we first import the smbus module, which provides an interface for using the I2C protocol on a Raspberry Pi. We then define the I2C bus number and the addresses of the master and slave devices.

We create an instance of the SMBus class with the appropriate bus number to initialize the I2C bus. We define four functions to perform I2C read and write operations: write\_byte (), read\_byte (), write\_bytes (), and read\_bytes (). These functions use the write\_byte (), read\_byte (), write\_i2c\_block\_data (), and read\_i2c\_block\_data () methods of the SMBus class, respectively.

We then write a byte to the master device using the write\_byte () method, and read a byte from the master device using the read\_byte () method. We print the received data in hexadecimal format.

Next, we write a sequence of bytes to the slave device using the write\_bytes () method, and read a sequence of bytes from the slave device using the read\_bytes () method. We print the received data as a list of bytes.

Note that the timing delays between I2C transactions are important to ensure reliable communication. The time. sleep () calls in the write\_byte () and write\_bytes () functions ensure that there is enough time between write operations to prevent the slave device from missing any data.

## Applications:

The I2C Protocol used to connect a maximum of 128 devices that are all connected to communicate with the SCL and SDL lines of the Master unit as well as slave devices. It supports Multi Master communication, which means two masters are used to communicate the external devices.

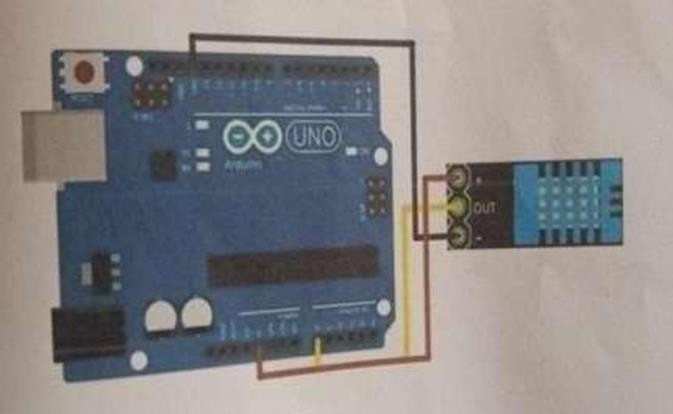
## Conclusion:

The I2C Protocol allows for each enabled device to have its own unique address, and as both master and slave devices to take turns communicating over a single line, it is possible for your Arduino or Genuino Board to communicate (in turn) with many devices, or other boards, while using just two pins of micro -controller.

# PROGRAM 5

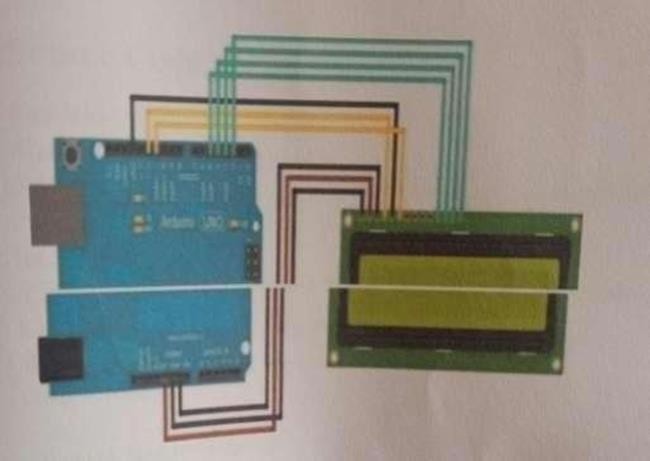
## Reading Temperature and Relative Humidity Value from the sensor Components Required:

Arduino UNO

DHT11 Temperature and Humidity Sensor Data

## Connections:

|  |  |
| --- | --- |
| Arduino Pins | DHT11 Sensor |
| 5V | +VE |
| A0 | OUT |
| GND | VE |



**Procedure:**

* 1. Set up pin connections DHTI Humidity Temperature sensor and Arduino UNO shown in figure.
  2. Open the Arduino UNO IDE and create and save a sketch source code as follows
  3. Compile sketch.
  4. Upload sketch on to Arduino UNO.

## Program:

Here's an example program in Python for reading temperature and relative humidity values from a DHT11 sensor using the Ada fruit\_DHT library:

import Adafruit\_DHT import time

# Define the sensor type and pin number sensor = Adafruit\_DHT.DHT11

pin = 4

# Loop to read sensor values and print them while True:

# Attempt to read the sensor values

humidity, temperature = Adafruit\_DHT.read\_retry (sensor, pin) # If successful, print the values

if humidity is not None and temperature is not None print ('Temperature: {0:.1f} C’. format(temperature)) print ('Humidity: {0:.1f} %’. format(humidity))

else:

print ('Failed to read sensor data')

# Wait for 2 seconds before reading the sensor again time. sleep (2)

In this code, we first import the Adafruit\_DHT library, which provides a high-level interface for reading temperature and humidity values from DHT11 and DHT22 sensors. We define the sensor type as DHT11 and the pin number as 4.

We then enter an infinite loop to continuously read and print the sensor values. We attempt to read the sensor values using the Adafruit\_DHT.read\_retry () function, which attempts to read the sensor several times until a successful reading is obtained. If the reading is successful, we print the temperature and humidity values in Celsius and percent, respectively. If the reading fails, we print an error message.

Finally, we wait for 2 seconds before reading the sensor again to avoid reading the sensor too frequently and causing measurement errors.

## Output:

The output of the program will vary depending on the temperature and humidity readings from the DHT11 sensor. Here is an example output:

Temperature: 22.0 C

Humidity: 42.0 %

Temperature: 22.0 C

Humidity: 41.0 %

Temperature: 21.0 C

Humidity: 40.0 %

The program continuously reads the temperature and humidity values every 2 seconds, and prints them to the console. In this example, the temperature values range from 21 to 22 degrees Celsius, and the humidity values range from 40 to 42 percent. The output will continuously update as the sensor values change.

## Applications:

A humidity sensor senses, measures and reports both moisture and air temperature. The ratio of moisture in the air to the highest amount of moisture at a particular air temperature is called relative humidity. Relative humidity becomes an important factor, when looking for comfort Automated humidity control systems use humidity sensors to test humidity levels and can control processes, such as chemical dryers, required to maintain a specific relative humidity in industries.

## Conclusion:

Air temperature and humidity sensors provide the benefit of obtaining measurement data for two parameters humidity and temperature using a single sensor. Sensor data help us to study and measure the humidity and temperature.

# OUTCOME

Working through these programming exercises is a great way to gain practical experience in working with various communication protocols and sensors. For example, the UART protocol program teaches you how to establish a serial communication between two devices and transmit data between them.

With the Point-To-Point Communication of Two Motes over the Radio Frequency program, you learn how to enable two wireless modules to communicate with each other via radio frequency signals. The Multi Point to Single point communication of Motes over the Radio Frequency program expands upon this concept, allowing multiple modules to communicate with a central device over radio frequency.

Meanwhile, the I2C Protocol program teaches you how to enable multiple devices to communicate with each other using a two-wire interface. Lastly, the Reading Temperature and Relative Humidity Value from the sensor program shows you how to read temperature and humidity values from a DHT11 sensor using the Adafruit\_DHT library.

By working through these exercises, you can gain knowledge and experience in programming for embedded systems, which is a valuable skillset in many industries.

For instance, you can use this knowledge to build home automation systems, smart sensors, and Internet of Things (IoT) devices. Moreover, you can develop skills that are highly sought after by employers in the embedded systems development field, which can lead to career opportunities.

Overall, completing these programming exercises can provide you with a solid foundation in working with different communication protocols and sensors, which can be useful for various projects and career paths.

# CONCLUSION

Working on these programming projects has been a valuable experience in gaining practical knowledge and skills in working with various communication protocols and sensors. By completing these exercises, I was able to develop my programming skills for embedded systems and learn how to establish communication between different devices.

The UART protocol program taught me how to establish serial communication between two devices and transmit data between them. The Point-To-Point Communication of Two Motes over the Radio Frequency program helped me understand how to use radio frequency signals to enable two wireless modules to communicate with each other.

With the Multi Point to Single point communication of Motes over the Radio Frequency program, I learned how to enable multiple modules to communicate with a central device over radio frequency.

The I2C Protocol program was especially useful in teaching me how to enable multiple devices to communicate with each other using a two-wire interface.

Lastly, the Reading Temperature and Relative Humidity Value from the sensor program taught me how to read temperature and humidity values from a DHT11 sensor using the Adafruit\_DHT library.

Through completing these projects, I was able to gain knowledge and experience in programming for embedded systems, which is an invaluable skillset in today's technology industry.

These skills can be applied to various projects and devices, such as home automation systems, smart sensors, and IoT devices.

Additionally, the experience I gained in programming for embedded systems can help me pursue career opportunities in the field of embedded systems development.

Overall, completing these programming projects has provided me with a strong foundation in understanding different communication protocols and sensors, and how to use them to establish communication between different devices.

I look forward to continuing my exploration of advanced topics in embedded systems development, and further refining my skills in programming for embedded systems.

# REFERENCES

1. https://exploreembedded.com/wiki/UART
2. https:/[/www.pantechsolutions.n](http://www.pantechsolutions.net/how-to-receive-a-string-from-uart)e[t/how-to-receive-a-string-from-uart](http://www.pantechsolutions.net/how-to-receive-a-string-from-uart)
3. https://znotes.org/caie/as-level/computer-science-9618/theory/communication/
4. https:/[/www.ge](http://www.geeksforgeeks.org/i2c-communication-protocol/)e[ksforgeeks.org/i2c-communication-protocol/](http://www.geeksforgeeks.org/i2c-communication-protocol/)
5. Shaik Sahil Babua, Arnab Rahab, Utpal Biswasc, Mrinal Kanti Naskard “HelloMsgC: A Practical Implementation of Hello Message Protocol in Wireless Sensor Network” International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013
6. https:/[/www.engineersgarage.com/](http://www.engineersgarage.com/articles-arduino-dht11-humidity-temperature-)a[rticles-arduino-dht11-humidity-temperature-](http://www.engineersgarage.com/articles-arduino-dht11-humidity-temperature-) sensor- interfacing/
7. Y. Chen, et al., "Design of Temperature and Humidity Monitoring System Based on the Internet of Things," 2019 4th International Conference on Electrical and Information Technologies (ICEIT), 2019, pp. 402-406. doi: 10.1109/ICEIT.2019.8862373.