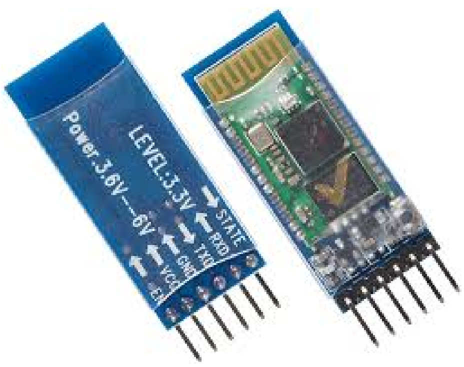
INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

PROJECT
ANALOG COMMUNICATION

TOPIC : ARDUINO BASED 2-WAY COMMUNICATION SYSTEM USING BLUETOOTH MODULES



SUBMITTED TO:
DR. PRASANT KUMAR SAHU

SUBMITTED BY:
ANIMESG PRASAND SINGH
ADITYA DUBEY
DEVEN SANJAY PATIL
NIKITA VERMA

Contents

Chapter 1

Introduction

1.1 Objective

The main objective of this project is to design and implement a 2-way communication system using Arduino microcontrollers and Bluetooth modules. The aim is to create a reliable and efficient wireless communication link between two Arduino-based devices, allowing them to exchange data seamlessly.
By achieving this objective, the project aims to demonstrate the feasibility of using Arduino and Bluetooth technology for developing custom wireless communication solution.

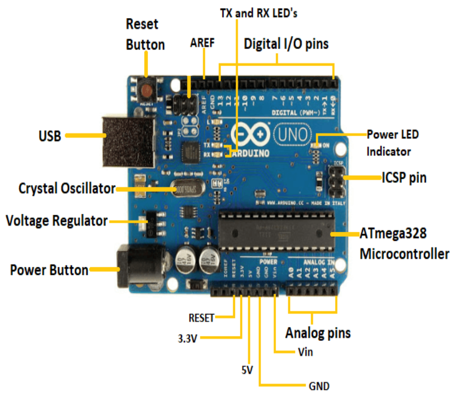
1.2 Overview of Bluetooth technology

Bluetooth technology is a wireless communication standard designed for short-range communication between devices. It operates in the 2.4 GHz frequency band and uses a technique called frequency-hopping spread spectrum (FHSS) to minimize interference and ensure reliable communication.
Bluetooth technology offers several key features and capabilities,
Pairing and Connection: Devices can be paired to establish a secure connection, allowing them to communicate with each other.
Profiles: Bluetooth defines various profiles that specify how different types of devices should communicate. Common profiles include Serial Port Profile (SPP) for data exchange, Advanced Audio Distribution Profile (A2DP) for audio streaming, and Hands-Free Profile (HFP) for hands-free calling.
Low Energy: Bluetooth Low Energy (BLE) is a power-efficient version of Bluetooth designed for applications requiring low power consumption, such as wearable devices and IoT applications.
Range: Bluetooth typically offers a range of up to 10 meters (30 feet), although this can vary depending on the environment and specific devices used.
Security: Bluetooth provides built-in security features, such as encryption and authentication, to ensure data privacy and protect against unauthorized access.

Chapter 2 Hardware Requirements

2.1 Arduino board

The Arduino board serves as the central component of the hardware setup, responsible for controlling the Bluetooth module, handling data exchange, and executing the programmed logic. Arduino Uno is one of the most commonly used Arduino boards due to its affordability, versatility, and extensive community support.



Features: Arduino Uno features a microcontroller (ATmega328P), digital and analog input/output pins, USB interface for programming and power supply, and onboard voltage regulator.
Compatibility: Arduino Uno is compatible with a wide range of shields and modules, making it easy to expand its functionalities by adding additional components such as sensors, displays, and communication modules.
Programming: Arduino Uno can be programmed using the Arduino IDE, a user-friendly development environment that simplifies the coding process, allowing even beginners to write and upload code to the board effortlessly.

2.2 Bluetooth modules (e.g., HC-05, HC-06)

Bluetooth modules enable wireless communication between the Arduino board and other devices, facilitating data exchange over short distances

.

HC-05: HC-05 is a Bluetooth Classic module known for its ease of use, affordability, and compatibility with Arduino. It supports Serial Port Profile (SPP) and is widely used for establishing wireless serial communication between devices.
HC-06: Similar to HC-05, HC-06 is another Bluetooth Classic module suitable for simple wireless communication tasks. It offers basic functionalities and is easy to integrate with Arduino projects

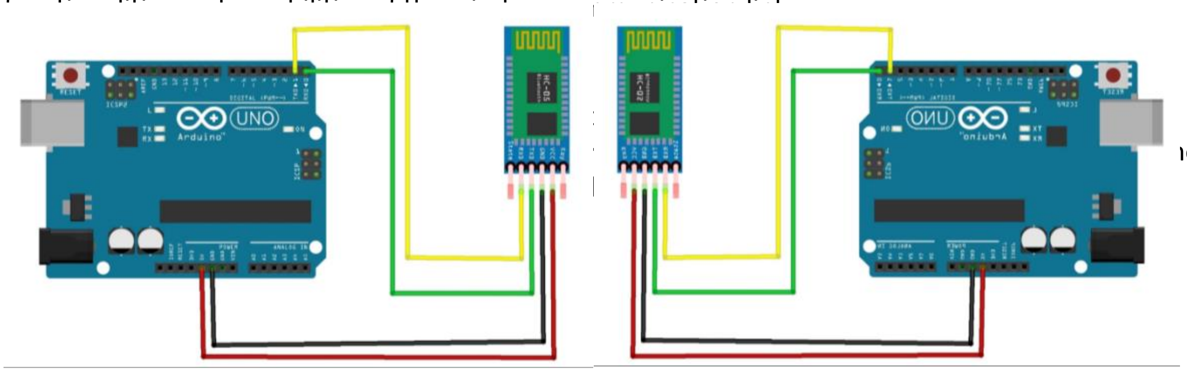
Chapter 3 :System Design

3.1 Block diagram

The block diagram provides a visual representation of the system architecture, illustrating the interaction between the main components of the Arduino-based 2-way communication system using Bluetooth modules. At the core of the system is the Arduino board, which serves as the central processing unit. Connected to the Arduino board is the Bluetooth module, facilitating wireless communication between the Arduino board and other devices. A power supply provides the necessary electrical power to the Arduino board and Bluetooth module, ensuring smooth operation of the system. Additional components such as LEDs, resistors, and push buttons may be integrated to enhance the functionality and user interface of the system, as depicted in the block diagram.

3.2 Explanation of each component

Arduino Board: The Arduino board acts as the brain of the system, responsible for controlling the Bluetooth module, executing the programmed logic, and managing data exchange between devices. It interfaces with the Bluetooth module through serial communication, sending and receiving data packets to establish and maintain the wireless connection.
Bluetooth Module (e.g., HC-05, HC-06): The Bluetooth module enables wireless communication between the Arduino board and other devices, facilitating data exchange over short distances. It operates based on the Bluetooth protocol, establishing a secure and reliable connection with compatible devices. The selected Bluetooth module (e.g., HC-05 or HC-06) determines the specific [illegible]



MASTER MODULE SLAVE MODULE

The circuit diagram provides a detailed schematic representation of the hardware connections between the Arduino board, Bluetooth module, power supply, and additional components. It illustrates the wiring configuration, pin connections, and component placement necessary to assemble the hardware setup accurately. Each component is represented by its symbol, with lines indicating the connections and pathways for electrical signals and power supply. The circuit diagram serves as a guide for building the physical circuit, ensuring correct connections and proper operation of the Arduino-based 2-way communication system using Bluetooth modules.

3.4 Description of communication protocol

The communication protocol defines the rules and conventions governing the exchange of data between the Arduino board and Bluetooth module, ensuring seamless and reliable wireless communication. In the context of this system, the Serial Port Profile (SPP) is commonly used for establishing a virtual serial port over Bluetooth, enabling transparent data exchange between devices. The protocol involves several key steps, including:
Initialization: The Arduino board and Bluetooth module are initialized and configured to establish a wireless connection. This involves setting the baud rate, pairing mode, and other parameters to

ensure compatibility and secure communication.
Data Exchange: Once the connection is established, the Arduino board and Bluetooth module can exchange data packets using serial communication. The Arduino board sends commands or messages to the Bluetooth module, which then transmits them wirelessly to the paired device. Similarly, data received wirelessly by the Bluetooth module is forwarded to the Arduino board for processing and interpretation.
Error Handling: The protocol includes mechanisms for error detection and recovery to ensure data integrity and reliability. Checksums or error-checking algorithms may be employed to verify the accuracy of transmitted data and detect any transmission errors or data corruption. In case of errors, retransmission or error correction techniques may be used to resolve the issue and maintain the integrity of the communication link.

Chapter 4 Implementation

The circuit diagram provides a detailed schematic representation of the hardware connections between the Arduino board, Bluetooth module, power supply, and additional components. It illustrates the wiring configuration, pin connections, and component placement necessary to assemble the hardware setup accurately. Each component is represented by its symbol, with lines indicating the connections and pathways for electrical signals and power supply. The circuit diagram serves as a guide for building the physical circuit, ensuring correct connections and proper operation of the Arduino-based 2-way communication system using Bluetooth.

Chapter 5 Applications

Bluetooth technology offers versatile applications across various domains, including wireless audio, data transfer, location-based services (LBS), and more.

In wireless audio, Bluetooth facilitates connections between devices like headphones, speakers, car audio systems, and gaming headsets, providing users with the convenience of wire-free audio enjoyment. Additionally, Bluetooth enables seamless data transfer between devices, allowing for file sharing, mobile device synchronization, peripheral connectivity, internet tethering, IoT device communication, and the establishment of wireless sensor networks.

In LBS, Bluetooth technology, particularly Bluetooth Low Energy (BLE) beacons, supports indoor navigation, proximity marketing, contact tracing, personalized experiences, asset tracking, and access control/authentication systems. These applications enhance user experiences, improve business operations, and contribute to public health efforts.

Overall, Bluetooth technology continues to revolutionize connectivity solutions, offering convenience, flexibility, and efficiency across various industries and domains.
Chapter 6 Challenges and Solutions

During the implementation of the project aimed at establishing serial communication between two Arduino boards using Bluetooth modules, several challenges were encountered. This report outlines the challenges faced and the corresponding solutions employed to overcome them.

1. Pairing Bluetooth Modules:

One of the primary challenges encountered during the project was the process of pairing the two Bluetooth modules, one configured as a master and the other as a slave. Pairing is essential for establishing a wireless connection between the modules and enabling data exchange.
Solution: The pairing process involved issuing AT commands to both modules to configure their roles, names, passwords, and communication parameters. However, ensuring successful pairing required

meticulous attention to detail and adherence to the correct syntax of AT commands. Additionally, verifying the physical connections between the modules and the Arduino boards proved crucial in ensuring seamless communication.

2. Baud Rate Mismatch:

Another significant challenge arose from a baud rate mismatch between the Bluetooth modules and the Arduino serial ports. Inconsistent baud rates can result in data corruption or loss during transmission, leading to communication errors.
Solution: Addressing the baud rate mismatch involved carefully configuring the baud rates of both the Bluetooth modules and the Arduino serial ports to ensure compatibility. Adjustments were made to the AT commands sent to the modules to set the desired baud rates, aligning them with the baud rates specified in the Arduino code. Thorough testing and troubleshooting were conducted to verify the stability and reliability of the communication link.

3. Serial Data Reception:

During the initial stages of the project, issues were encountered with the reception of serial data on the Arduino boards. Despite successful pairing and configuration of the Bluetooth modules, inconsistencies in data reception hindered the exchange of information between the master and slave boards.
Solution: Debugging the serial data reception involved careful examination of the code and identification of potential bottlenecks or errors. Techniques such as printing debug messages and monitoring serial output aided in pinpointing the source of the issue. Additionally, adjusting the software serial buffer size and implementing error-checking mechanisms helped improve the reliability of data reception, ultimately resolving the issue.
Chapter 7 Conclusion

The project aimed at establishing serial communication between two Arduino boards using Bluetooth modules encountered significant challenges, notably in pairing the modules and resolving baud rate mismatches. Despite these hurdles, successful bidirectional communication was achieved through meticulous troubleshooting and refinement. Notably, the project demonstrated success in exchanging text data between two laptops via the Arduino boards, showcasing the practical application of the wireless communication setup. This experience underscores the importance of perseverance and innovation in overcoming technical hurdles. Moving forward, the lessons learned will inform future projects in the realm of wireless communication and embedded systems, driving advancements in IoT and beyond.

Chapter8 Code snippets

Serial Communication using SoftwareSerial

```
#include <SoftwareSerial.h>
#define tx 8
#define rx 9
SoftwareSerial mySerial(rx, tx); //RX, TX
void setup()
{
  Serial.begin(9600);
  mySerial.begin(9600);

}
void loop()
{
  if(mySerial.available())
```

```
 {
   Serial.println(mySerial.readString());
 }
 if(Serial.available()){
   mySerial.write(Serial.read());
 }
}
```

Explanation:

This Arduino code snippet demonstrates serial communication between two devices using the SoftwareSerial library. The purpose of the code is to establish a communication link between the Arduino's hardware serial (Serial) and a software-based serial port (mySerial) defined on pins 8 (TX) and 9 (RX) of the Arduino board.

Setup Function:
In the setup() function:
Serial.begin(9600); initializes the hardware serial communication at a baud rate of 9600.
mySerial.begin(9600); initializes the software serial communication at the same baud rate of 9600.
Loop Function:
The loop() function continuously checks if there's any data available on the software serial port (mySerial.available()).
If data is available, it reads the data (mySerial.readString()) and prints it to the hardware serial port (Serial.println()), which is connected to the Serial Monitor on the Arduino IDE.
It also checks if there's any data available on the hardware serial port (Serial.available()). If data is available, it reads the data (Serial.read()) and writes it to the software serial port (mySerial.write()), which is connected to the other Arduino board

Slave Module Configuration with SoftwareSerial

```
#include <SoftwareSerial.h>

#define BT_NAME   "BT_Slave"

SoftwareSerial mySerial(8, 9); // RX, TX

void setup()
{
 Serial.begin(38400);

 mySerial.begin(38400);
 Serial.println("Arduino receiver");

 mySerial.print("AT\r\n");
 delay(200);
 mySerial.print("AT+RMAAD\r\n");
 delay(200);
 mySerial.print("AT+ADDR?\r\n");
 delay(200);

 mySerial.print("AT+NAME="+String(BT_NAME)+"\r\n");
 delay(200);
 mySerial.print("AT+PSWD=\"1234\"\r\n");
```

```
  delay(200);
  mySerial.print("AT+ROLE?\r\n");
  delay(200);
  mySerial.print("AT+UART=9600,0,0\r\n");
  delay(500);
}

void loop()
{
}
```

Master Module Configuration with SoftwareSerial

```
#include <SoftwareSerial.h>

#define BT_NAME   "BT_Master"

#define SLAVE_ADDRESS "0000,13,0AA5EB"  //0:13:AA5EB

SoftwareSerial mySerial(2, 3); // RX, TX

void setup()
{
  Serial.begin(38400);

  mySerial.begin(38400);
  Serial.println("Arduino Sender");

  mySerial.print("AT\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+RMAAD\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+ROLE=1\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+NAME="+String(BT_NAME)+"\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+PSWD=\"1234\"\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+BIND="+String(SLAVE_ADDRESS)+"\r\n");
  updateSerial();
  delay(200);
  mySerial.print("AT+UART=38400,0,0\r\n");
  updateSerial();
  delay(500);
  mySerial.print("AT+UART?\r\n");
  updateSerial();
  delay(200);
}
void loop()
{

}
```

Explanation:
This Arduino code is designed to configure and communicate with a Bluetooth module using AT commands via SoftwareSerial. The Bluetooth module is set up as a slave device, and the Arduino acts as a master to send AT commands for configuration. The SoftwareSerial library is used to establish a serial communication link between the Arduino and the Bluetooth module, with the Bluetooth module connected to pins 8 (TX) and 9 (RX) of the Arduino board.

Bluetooth Module Connection:
The Bluetooth module is connected to the Arduino as follows:
VCC (Bluetooth) to 5V (Arduino)
GND (Bluetooth) to GND (Arduino)
TX (Bluetooth) to 8 (Arduino)during the upload process and 9 (Arduino)during data transmission
RX (Bluetooth) to 9 (Arduino)during the upload process and 8 (Arduino)during data transmission

SLAVE CONFIGURATION:
AT: Checks if the Bluetooth module is responsive.
AT+RMAAD: Clears any existing paired devices.
AT+ROLE=0: Sets the Bluetooth module as a slave.
AT+NAME=BT_Slave: Sets the name of the Bluetooth module to "BT_Slave".
AT+PSWD="1234": Sets the pairing password to "1234".
AT+ADDR?: Queries the Bluetooth module for its address.
AT+UART=9600,0,0: Sets the UART communication parameters to 9600 baud rate, 0 stop bits, and no parity.

MASTER CONFIGURATION:
AT: Checks if the Bluetooth module is responsive.
AT+RMAAD: Clears any existing paired devices.
AT+ROLE=1: Sets the Bluetooth module as a master.
AT+NAME=BT_Master: Sets the name of the Bluetooth module to "BT_Master".
AT+PSWD="1234": Sets the pairing password to "1234".
AT+BIND=0000,13,0AA5EB: Binds the Bluetooth module to the specified slave address.
AT+UART=9600,0,0: Sets the UART communication parameters to 9600 baud rate, 0 stop bits, and no parity.
AT+UART?: Queries the current UART settings.

Bibliography

https://www.instructables.com/Arduino-Bluetooth-Master-and-Slave-Using-Any-HC-05/

https://www.hackster.io/my_engineeringStuffs/master-and-slave-two-way-bluetooth-data-communication-29538e
https://www.electronicsforu.com/electronics-projects/hardware-diy/arduino-chatting-diy
https://github.com/stechiez/Arduino/blob/master/HC05_master_Slave/hc05_master/hc05_master.ino
https://circuitdigest.com/microcontroller-projects/arduino-to-arduino-bluetooth-communication-using-master-slave-configuration