राष्ट्रीय प्रौद्योगिकी संस्थान गोवा

NATIONAL INSTITUTE OF TECHNOLOGY GOA

MINI PROJECT

“SMART BLUETOOTH CONTROLLED FAN”

NAME:

MANOBHAV SACHAN

DEEPAK SURESH

ROLL NO.:

20ECE1041

20ECE1013

SEMESTER: SIXTH

YEAR: THIRD

BRANCH: ECE

ACADEMIC SESSION:

JANUARY-MAY (2023)

INDEX

Sr.No.	Title	Page No.
1)	Abstract	4
2)	Introduction to the Simulation Part	5
3)	Modulation and Demodulation Part of Bluetooth	6
4)	MATLAB Code	7
5)	Output B.E.R. and P.E.R. Graphs	11
6)	Introduction to the Hardware Part	12
7)	Components Required	13
8)	Circuit Diagram	16
9)	Arduino Code	17
10)	Mobile App Interface	18
11)	Results	19
12)	Application and Future Scopes	20
13)	Conclusions	21

ABSTRACT

The aim of this project was to create a smart Bluetooth-controlled fan. The project consisted of two main parts: a Bluetooth receiver and transmitter made using MATLAB, and a hardware component using an HC05 chip for Bluetooth modulation and demodulation with an Arduino board and Motor driver Chip. The resulting system allows for easy and efficient control of the fan's speed and direction via a mobile app.

To achieve this, standard Bluetooth modulation and demodulation techniques were used in the MATLAB software. For the hardware component, an H-bridge motor driver and PWM technique were employed to provide smooth and precise control of the fan's speed and direction. The use of a Bluetooth module and mobile app provided a wireless means of controlling the fan, which could be useful in situations where wired connections are impractical or unsafe.

The development and implementation of the software were carried out using the Arduino board, which provided a reliable and user-friendly platform. The MIT App Inventor was also used to facilitate the rapid development of the mobile app, requiring minimal programming skills.

This project demonstrated the potential of combining off-the-shelf hardware and software to develop innovative and useful systems. The resulting system has potential applications in various fields, including robotics, automation, and home appliances. For example, it could be used to control the speed and direction of a conveyor belt in a manufacturing plant or regulate the rotation of a fan in a ventilation system.

Future improvements could include adding features such as motor current monitoring, temperature monitoring, and overload protection to make the system more efficient and safer. Overall, this project successfully demonstrated the development of a system for controlling the speed and direction of a DC motor using Bluetooth through a mobile app.

Introduction to Simulation Part

Bluetooth is a short-range Wireless Personal Area Network (WPAN) technology, operating in the globally unlicensed industrial, scientific, and medical (ISM) band in the frequency range of 2.4 GHz to 2.485 GHz. In Bluetooth technology, data is divided into packets. Each packet is transmitted on one of the 79 designated Bluetooth channels. The bandwidth of each channel is 1 MHz. Bluetooth implements the frequency-hopping spread spectrum (FHSS) scheme to switch a carrier between multiple frequency channels by using a pseudorandom sequence known to the transmitter and the receiver.

The Bluetooth standard specifies these PHY modes:

Basic rate (BR) - Mandatory mode, uses Gaussian frequency shift keying (GFSK) modulation with a data rate of 1 Mbps.

Enhanced data rate (EDR) - Optional mode, uses phase shift keying (PSK) modulation with these two variants:

- EDR2M: Uses pi/4-DQPSK with a data rate of 2 Mbps
- EDR3M: Uses 8-DPSK with a data rate of 3 Mbps

This end-to-end Bluetooth BR/EDR PHY simulation determines BER and PER performance of one Bluetooth packet that has RF impairments and AWGN. Each packet is generated over a loop of a vector equal to length of the energy-to-noise density ratio (Eb/No) using **the bluetoothWaveformGenerator** function by configuring the **bluetoothWaveformConfig** object.

To accumulate the error rate statistics, the generated waveform is altered with RF impairments and AWGN before passing through the receiver.

These RF impairments are used to distort the packet:

- DC offset
- Carrier frequency offset
- Static timing offset
- Timing drift

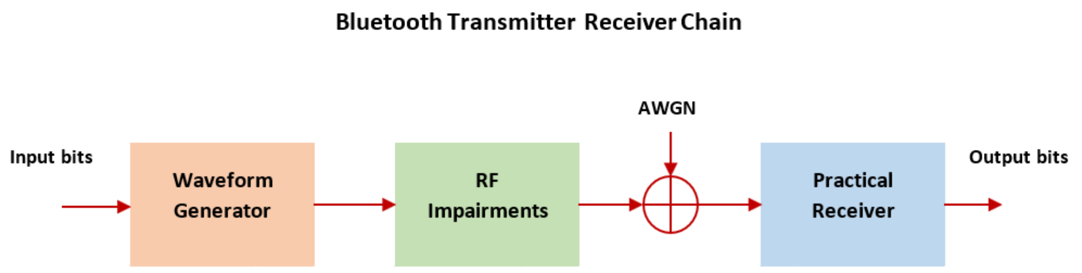
White Gaussian noise is added to the generated Bluetooth BR/EDR waveforms. The distorted and noisy waveforms are processed through a practical Bluetooth receiver performing these operations:

- Remove DC offset
- Detect the signal bursts
- Perform matched filtering
- Estimate and correct the timing offset
- Estimate and correct the carrier frequency offset
- Demodulate BR/EDR waveform

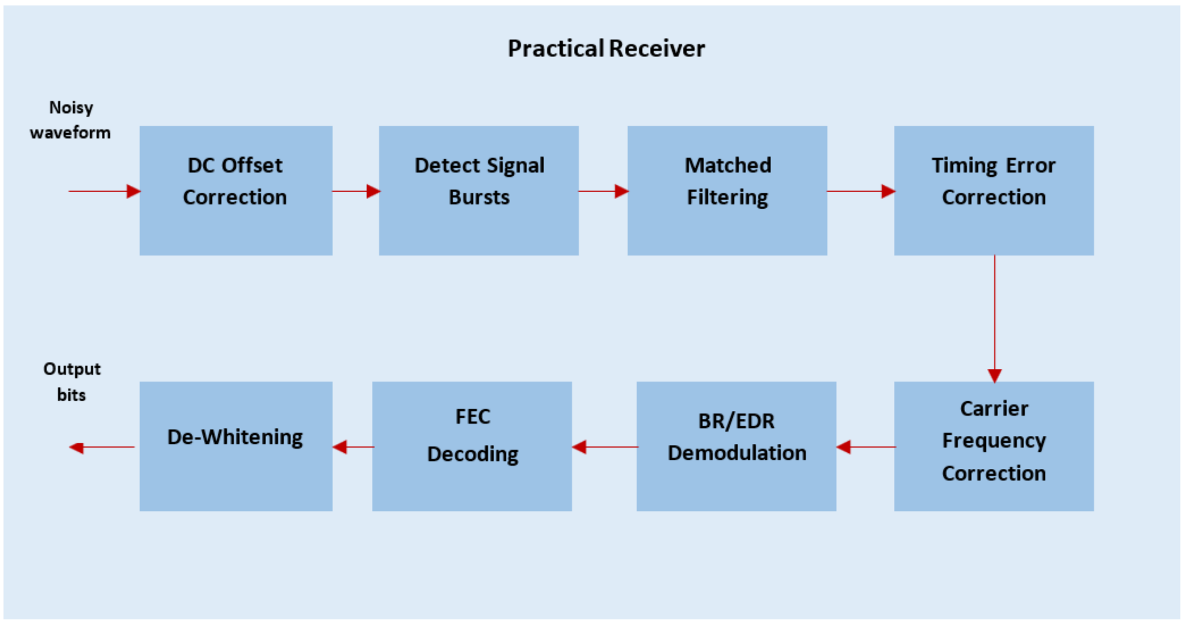
- **Perform forward error correction (FEC) decoding**
- **Perform data de-whitening**
- **Perform header error check (HEC) and cyclic redundancy check (CRC)**
- **Outputs decoded bits and decoded packet statistics based on decoded lower address part (LAP), HEC, and CRC.**

Modulation and Demodulation Part of Bluetooth

This block diagram illustrates the processing steps for each Bluetooth BR/EDR PHY packet.



To determine the BER and the PER, compare the recovered output bits with the transmitted data bits.



MATLAB Code

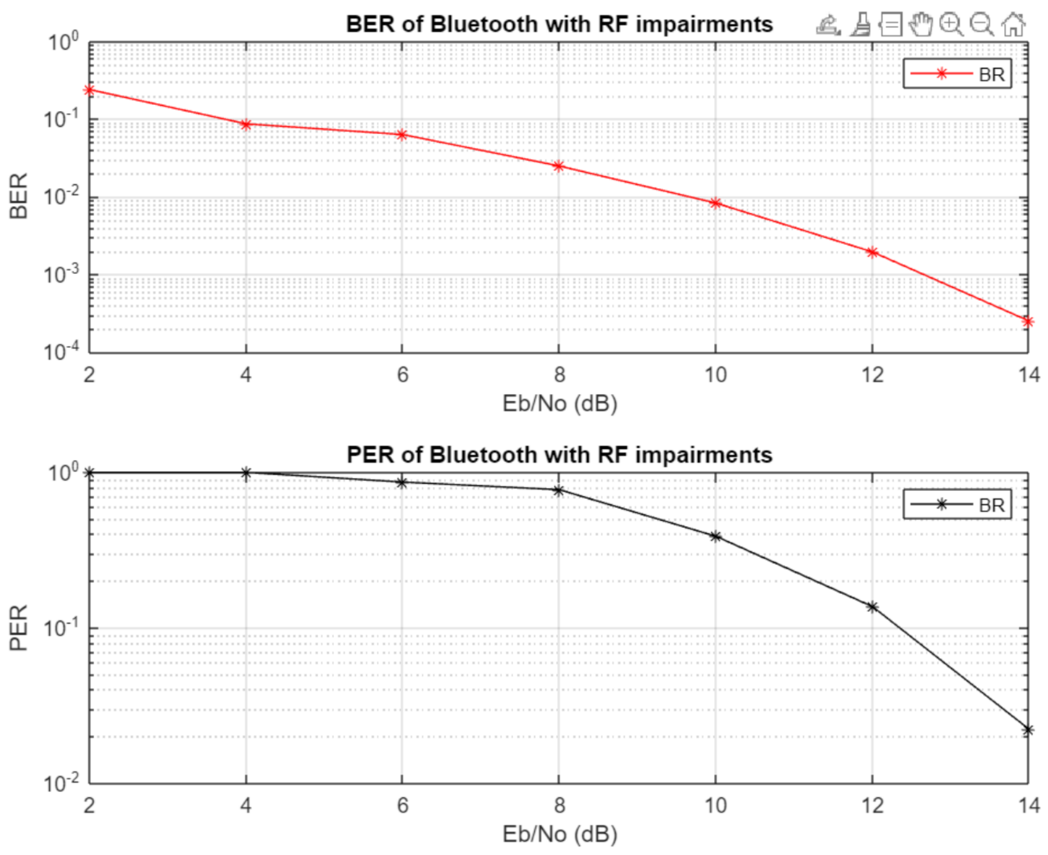
```
% Eb/No in dB
EbNo = 2:2:14;
% Maximum number of bit errors simulated at each Eb/No
point
maxNumErrs = 100;
% Maximum number of bits accumulated at each Eb/No point
maxNumBits = 1e6;
% Maximum number of packets considered at each Eb/No
point
maxNumPkts = 1000;


phyMode = 'BR';       % PHY transmission mode
bluetoothPacket = 'FHS';              % Type of Bluetooth
packet, this value can be: {'ID',
                                        %
'NULL','POLL','FHS','HV1','HV2','HV3','DV','EV3',
                                        %
'EV4','EV5','AUX1','DM3','DM1','DH1','DM5','DH3',
                                        % 'DH5','2-DH1','2-
DH3','2-DH5','2-DH1','2-DH3',
                                        %  '2-DH5','2-
EV3','2-EV5','3-EV3','3-EV5'}
sps = 8;                                % Samples per
symbol, must be greater than 1


frequencyOffset = 6000;% In Hz
timingOffset = 0.5;    % In samples, less than 1
microsecond
timingDrift = 2;        % In parts per million
dcOffset = 2;            % Percentage w.r.t maximum
amplitude value
symbolRate = 1e6;                                  % Symbol
Rate

% Create timing offset object
timingDelayObj = dsp.VariableFractionalDelay;

% Create frequency offset object
frequencyDelay =
comm.PhaseFrequencyOffset('SampleRate',symbolRate*sps);
```

```
ber = zeros(1,length(EbNo));        % BER results
per = zeros(1,length(EbNo));        % PER results
bitsPerByte = 8;                    % Number of bits per
byte
% Set code rate based on packet
if
any(strcmp(bluetoothPacket,{'FHS','DM1','DM3','DM5','HV2'
,'DV','EV4'}))
    codeRate = 2/3;
elseif strcmp(bluetoothPacket,'HV1')
    codeRate = 1/3;
else
    codeRate = 1;
end
% Set number of bits per symbol based on the PHY
transmission mode
bitsPerSymbol = 1+ (strcmp(phyMode,'EDR2M'))*1
+(strcmp(phyMode,'EDR3M'))*2;

% Get SNR from EbNo values
snr = EbNo + 10*log10(codeRate) + 10*log10(bitsPerSymbol)
- 10*log10(sps);
% Create a Bluetooth BR/EDR waveform configuration object
txCfg =
bluetoothWaveformConfig('Mode',phyMode,'PacketType',bluet
oothPacket,...
                                'SamplesPerSymbol',sps);
if strcmp(bluetoothPacket,'DM1')
    txCfg.PayloadLength = 17; % Maximum length of DM1
packets in bytes
end
dataLen = getPayloadLength(txCfg);  % Length of the
payload
% Get PHY properties
rxCfg = getPhyConfigProperties(txCfg);

for iSnr = 1:length(snr)
        rng default
        % Initialize error computation parameters
        errorCalc = comm.ErrorRate;
        berVec = zeros(3,1);
        pktCount = 0;  % Counter for number of detected
Bluetooth packets
        loopCount = 0; % Counter for number of packets at
each SNR value
        pktErr = 0;
```

```
            while((berVec(2) < maxNumErrs) && (berVec(3) <
maxNumBits) && (loopCount < maxNumPkts))
                txBits = randi([0 1],dataLen*bitsPerByte,1);
% Data bits generation
                txWaveform =
bluetoothWaveformGenerator(txBits,txCfg);

                % Add Frequency Offset
                frequencyDelay.FrequencyOffset =
frequencyOffset;
                transWaveformCFO =
frequencyDelay(txWaveform);

                % Add Timing Delay
                timingDriftRate = (timingDrift*1e-
6)/(length(txWaveform)*sps);% Timing drift rate
                timingDriftVal =
timingDriftRate*(0:1:(length(txWaveform)-1))';% Timing
drift
                timingDelay =
(timingOffset*sps)+timingDriftVal;   % Static timing
offset and timing drift
                transWaveformTimingCFO =
timingDelayObj(transWaveformCFO,timingDelay);

                % Add DC Offset
                dcValue =
(dcOffset/100)*max(transWaveformTimingCFO);
                txImpairedWaveform = transWaveformTimingCFO +
dcValue;

                 % Add AWGN
                txNoisyWaveform =
awgn(txImpairedWaveform,snr(iSnr),'measured');

                % Receiver Module
                [rxBits,decodedInfo,pktStatus]...
                                            =
helperBluetoothPracticalReceiver(txNoisyWaveform,rxCfg);
                numOfSignals = length(pktStatus);
                pktCount = pktCount+numOfSignals;
                loopCount = loopCount+1;

                % BER and PER Calculations
                L1 = length(txBits);
                L2 = length(rxBits);
                L = min(L1,L2);
            if(~isempty(L))
               berVec =
errorCalc(txBits(1:L),rxBits(1:L));
            end
            pktErr = pktErr+sum(~pktStatus);
         end
         % Average of BER and PER
         per(iSnr) = pktErr/pktCount;
         ber(iSnr) = berVec(1);
         if ((ber(iSnr) == 0) && (per(iSnr) == 1))
             ber(iSnr) = 0.5; % If packet error rate is
1, consider average BER of 0.5
         end
         if
~any(strcmp(bluetoothPacket,{'ID','NULL','POLL'}))
             disp(['Mode ' phyMode ', '...
                 'Simulated for Eb/No = ',
num2str(EbNo(iSnr)), ' dB' ', '...
                 'obtained BER:',num2str(ber(iSnr)),'
obtained PER: ',...
                    num2str(per(iSnr))]);
         else
             disp(['Mode ' phyMode ', '...
                 'Simulated for Eb/No = ',
num2str(EbNo(iSnr)), ' dB' ', '...
                 'obtained PER: ',num2str(per(iSnr))]);
         end
end


figure,
if any(strcmp(bluetoothPacket,{'ID','NULL','POLL'}))
    numOfPlots = 1; % Plot only PER
else
    numOfPlots = 2; % Plot both BER and PER
    subplot(numOfPlots,1,1),semilogy(EbNo,ber.','-r*');
    xlabel('Eb/No (dB)');
    ylabel('BER');
    legend(phyMode);
    title('BER of Bluetooth with RF impairments');
    hold on;
    grid on;
end
subplot(numOfPlots,1,numOfPlots),semilogy(EbNo,per.','-
k*');
xlabel('Eb/No (dB)');
ylabel('PER');
```

```
legend(phyMode);
title('PER of Bluetooth with RF impairments');
hold on;
grid on;
```

Output BER and PER Graphs



Introduction to Hardware Part

Introduction: In this project, we will explore how to control a DC motor's speed and direction via Bluetooth through a mobile app. We will use pulse-width modulation (PWM) and direction control signals to adjust the motor's speed and direction, respectively. This project is not only fun and engaging but also provides an excellent opportunity to learn about microcontrollers, motor drivers, Bluetooth modules, and mobile app development.

Objectives: The primary objectives of this project are:

- **To build a circuit that controls the speed and direction of a DC motor using a microcontroller board and a motor driver IC.**
- **To establish a Bluetooth connection between the microcontroller board and a mobile device using a Bluetooth module.**
- **To develop a mobile app that can send commands to the microcontroller board to control the DC motor's speed and direction.**

Methodology: The project will involve the following steps:

- **Selecting a suitable DC motor and motor driver IC that can handle the motor's voltage and current requirements.**
- **Setting up the motor driver circuit and connecting the DC motor to the circuit.**
- **Choosing a microcontroller board and a Bluetooth module that are compatible with each other.**
- **Connecting the microcontroller board and the Bluetooth module to the motor driver circuit.**
- **Developing a mobile app that can send PWM and direction control signals to the microcontroller board.**
- **Writing the code for the microcontroller board to receive data from the Bluetooth module and control the DC motor's speed and direction.**
- **Testing the circuit by connecting the mobile app to the Bluetooth module and sending commands to the microcontroller board to control the DC motor.**

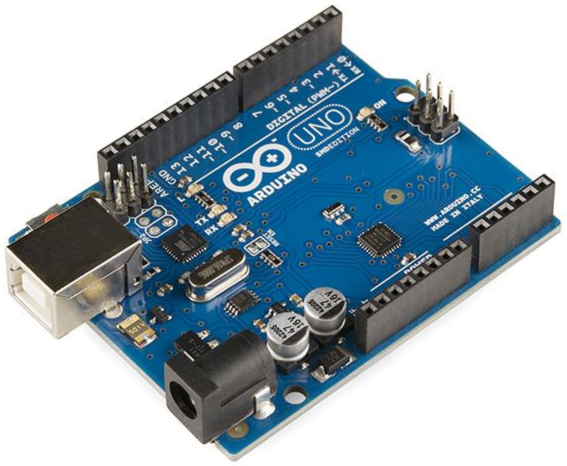
Expected Results: Upon completion of this project, we expect to have a functional circuit that can control the speed and direction of a DC motor using a mobile app via Bluetooth. We will also have gained practical knowledge of microcontrollers, motor drivers, Bluetooth modules, and mobile app development.

Conclusion: This project provides an excellent opportunity to learn about the basics of electronics, microcontrollers, and mobile app development. It also demonstrates how to control a DC motor's speed and direction using pulse-width modulation and direction control signals via Bluetooth. With this project, we hope to inspire others to explore the world of electronics and embedded systems.

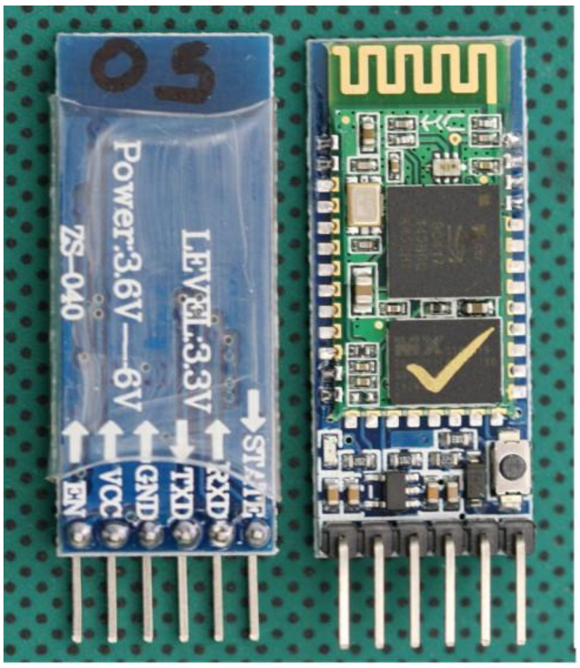
Components Required

We will need:

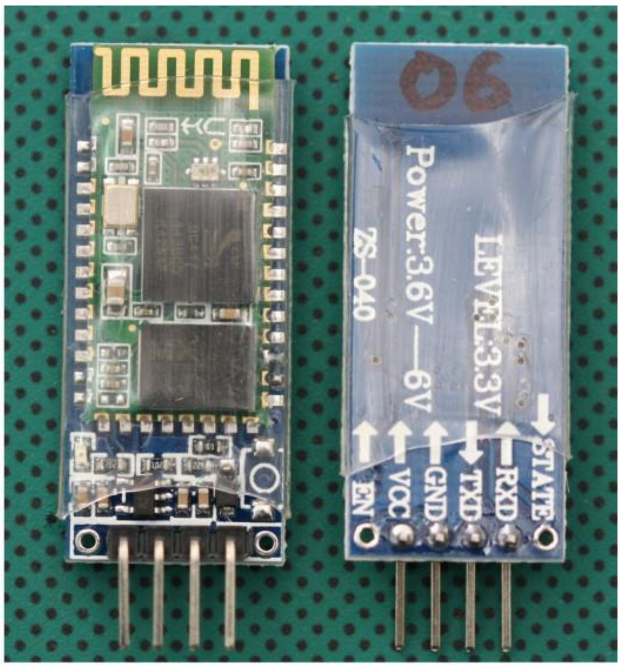
- **An Arduino Uno.**



- **A Bluetooth Low Energy (BLE) module, such as the HC-06 which is a slave only, or the HC-05 that can be a master or slave. Either will work because the module will be used as a slave. They also share the same four middle pins, which are the only ones we need.**

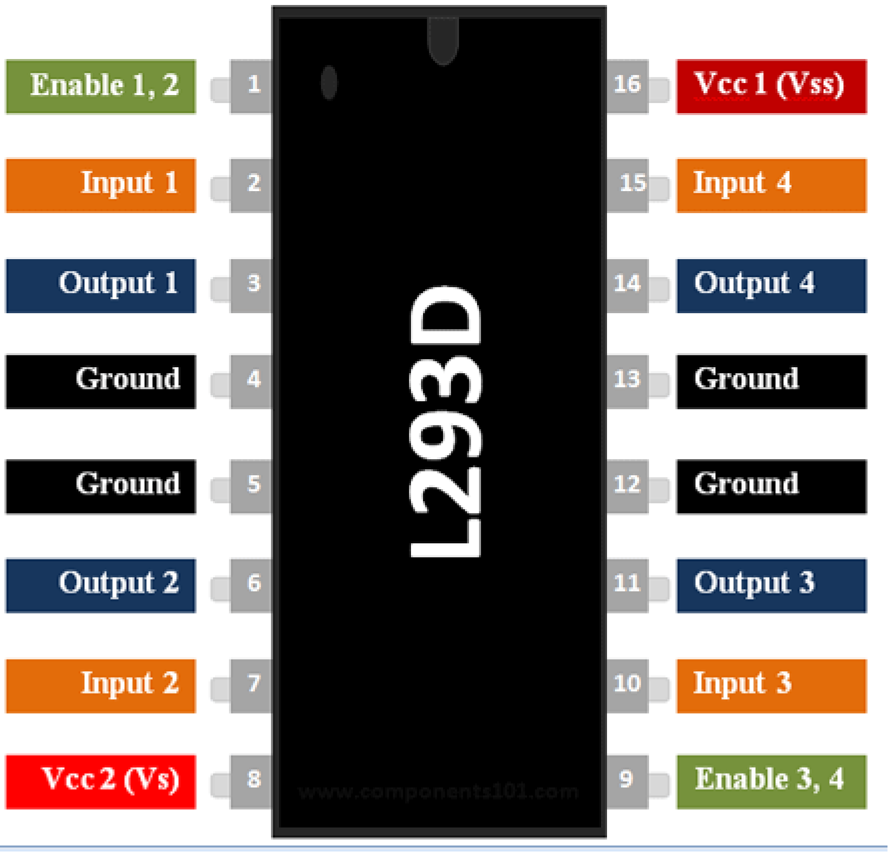


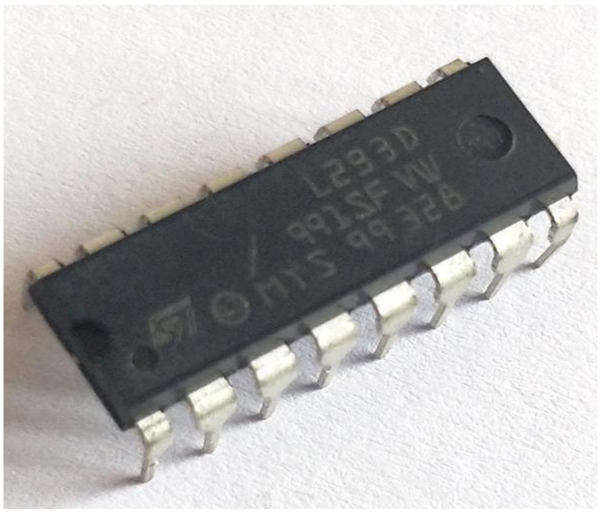
HC-05



HC-06

- **An L293D motor driver IC. This 16-pin dual H-bridge motor driver will allow us to control the spinning direction and speed of the motor.**



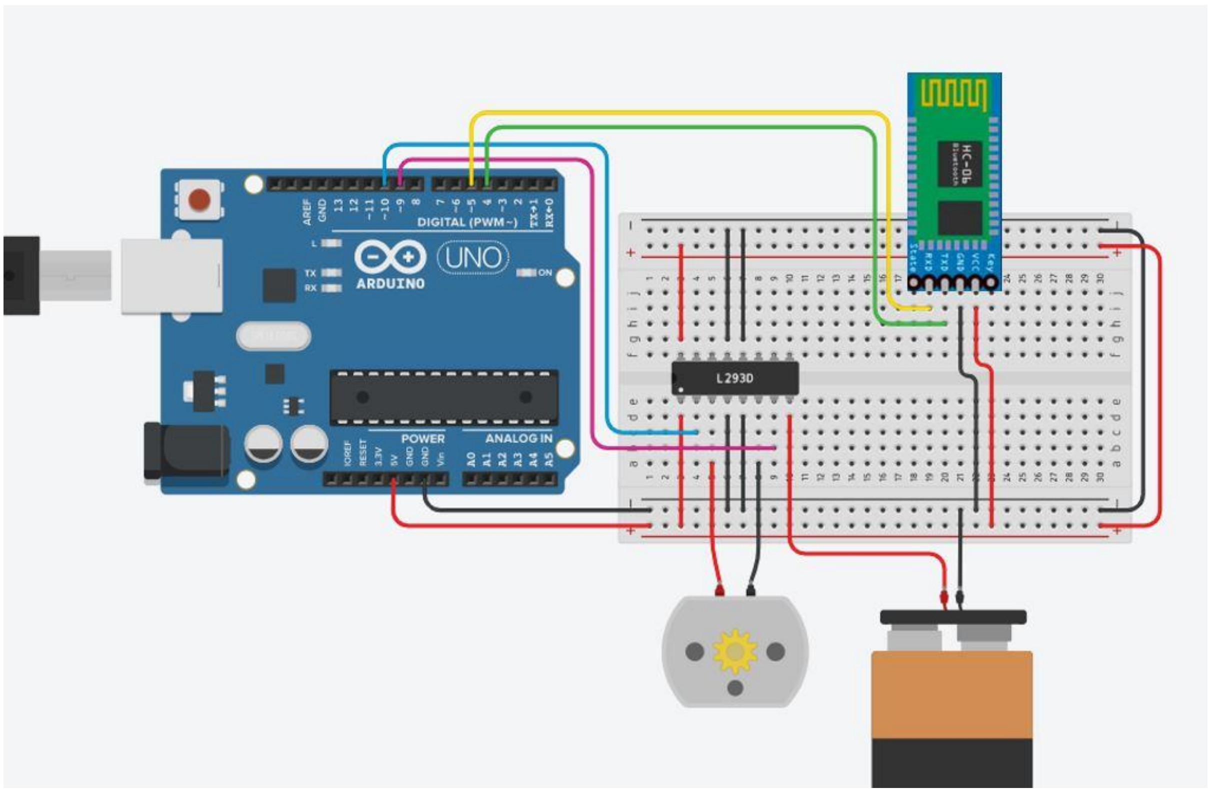


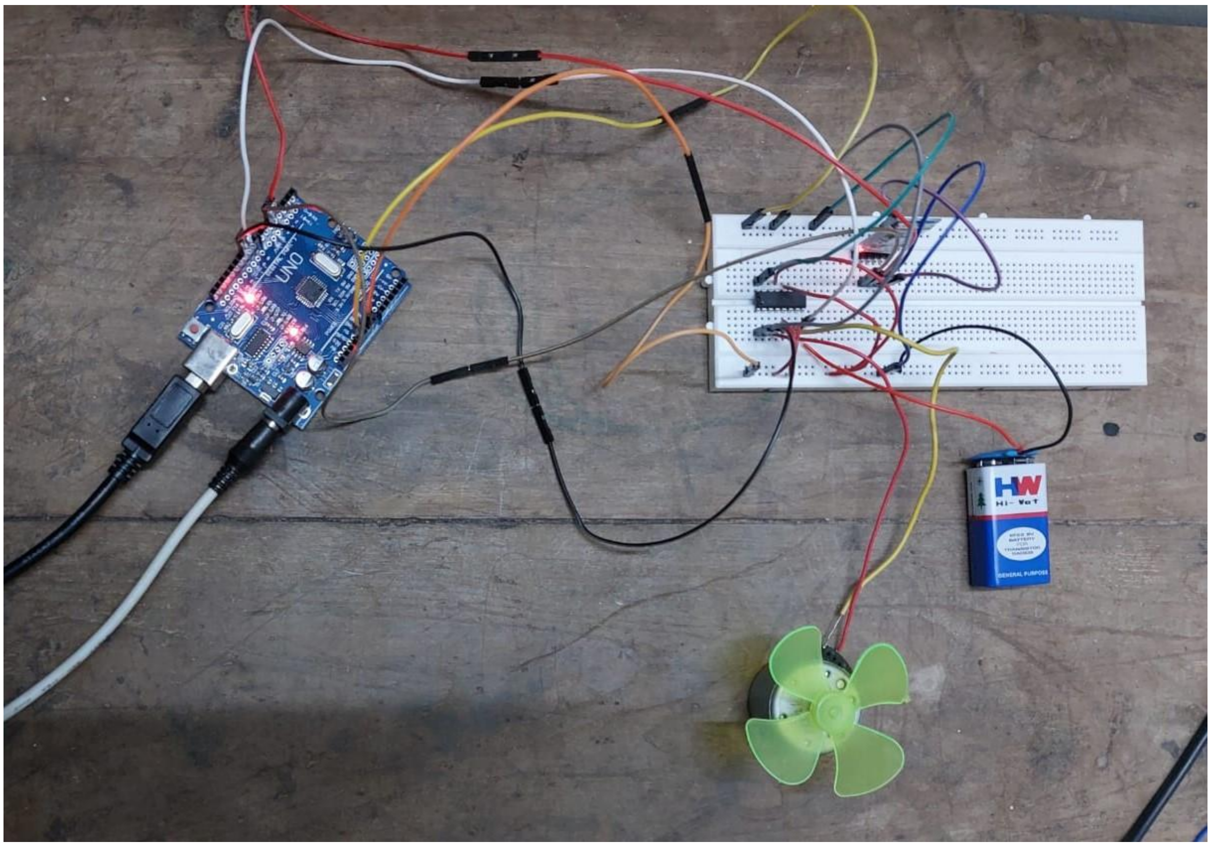
- A 12V DC motor.

- A 12V battery pack.

- A power source for the Arduino.
- Jumper wires and a breadboard.

Circuit Diagram





Arduino Code

```
#include <SoftwareSerial.h>
SoftwareSerial bt_ser(4, 5);  //connected  to RX and TX pins for
serial data communication
char c[6];
int i = 0, speed_value = 0, send_value;

#define pwm1 9   //input 2
#define pwm2 10  //input 1

boolean motor_dir = 0;

void setup() {
  Serial.begin(9600);
  bt_ser.begin(9600);
  pinMode(pwm1, OUTPUT);
  pinMode(pwm2, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {

  while (bt_ser.available())  //when data is transmitted
  {
    if (bt_ser.available() > 0) {
      c[i] = bt_ser.read();  //reading the string sent from
master  device
      Serial.print(c[i]);
      i++;
    }
    if (c[i - 1] == 'N')  //if button is pressed
    {
      motor_dir = !motor_dir;  //toggle  direction variable
      if (motor_dir)           //setting direction,  pwm1 and pwm2
are opposites
        digitalWrite(pwm2, 0), digitalWrite(LED_BUILTIN, HIGH);
      else
        digitalWrite(pwm1, 0), digitalWrite(LED_BUILTIN, LOW);
    }
    if (c[i - 1] == 'S') speed_value = 0;
    if (c[i - 1] == '1') speed_value = 70; // Slow Speed
    else if (c[i - 1] == '2') speed_value = 160; // Medium Speed
    else if (c[i - 1] == '3') speed_value = 250; // Fast Speed
  }

  //interpreting speed from string
  if (motor_dir)  //for a given direction
  {
    analogWrite(pwm1, speed_value);
  } else {  //for opposite direction
    analogWrite(pwm2, speed_value);
  }
  i = 0;
}
```

Mobile App Interface

We decided to use an application called Bluetooth Electronics that we found on the Play Store, because it gives the option to create dashboards as well as customizing the data the components on the dashboard send, or how they interact with each other.

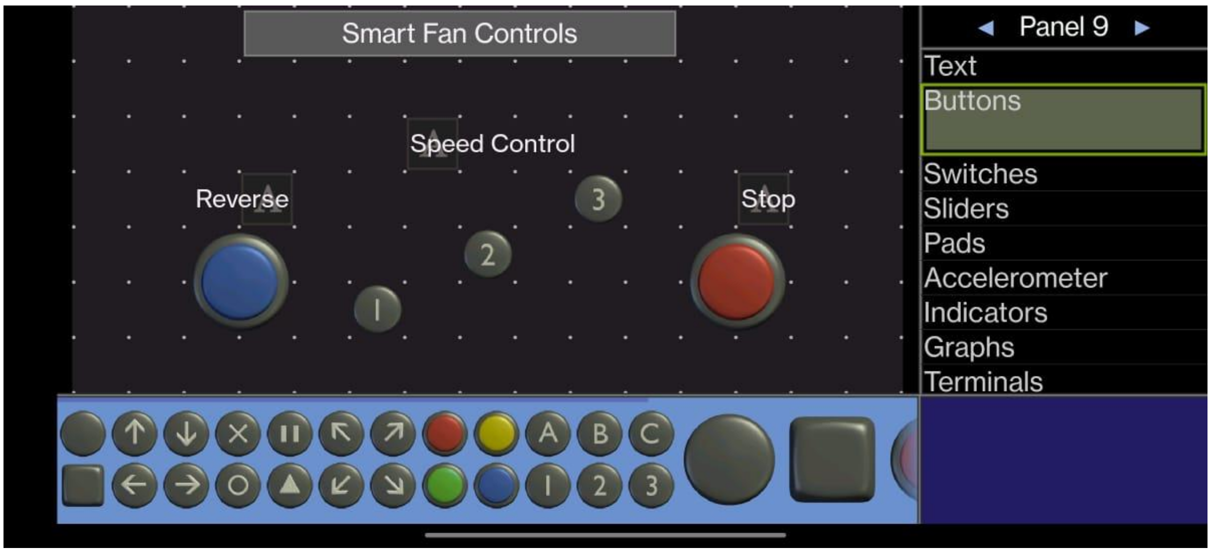
In order to create this interface, you will need to edit a new dashboard, and from the editing menu select a slider, a button and a text box.

The button should be edited as so (release text left as blank):

Press text: N, S, 1, 2, 3.

Release text:

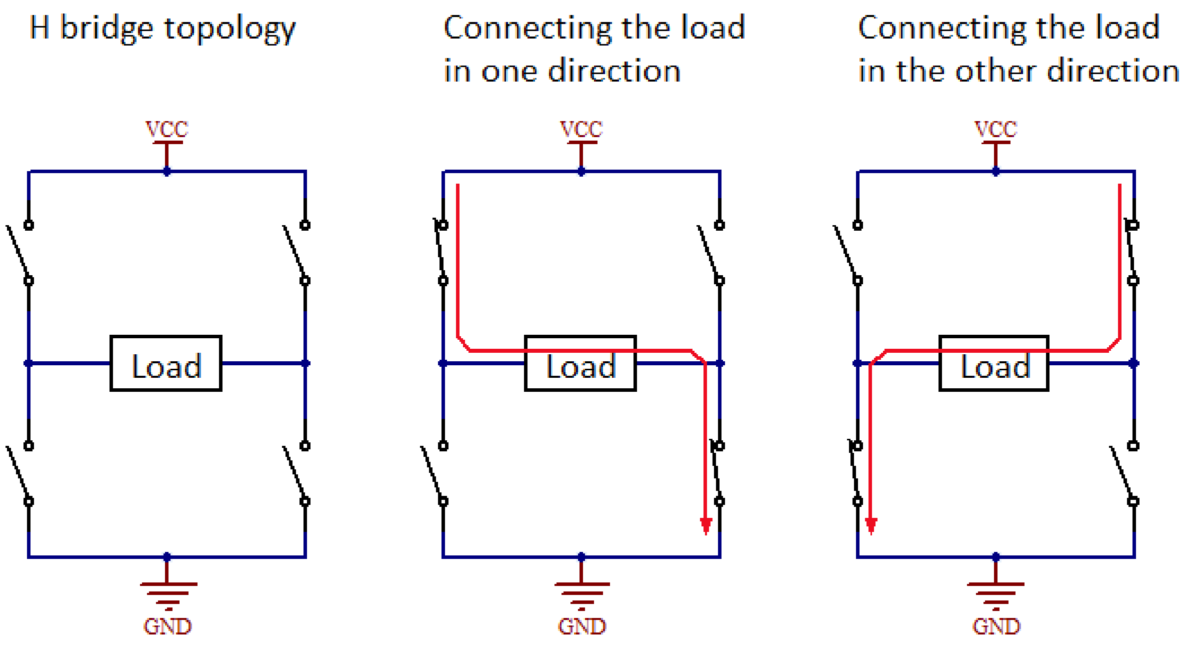
You can then add text to clarify the dashboard.



Results

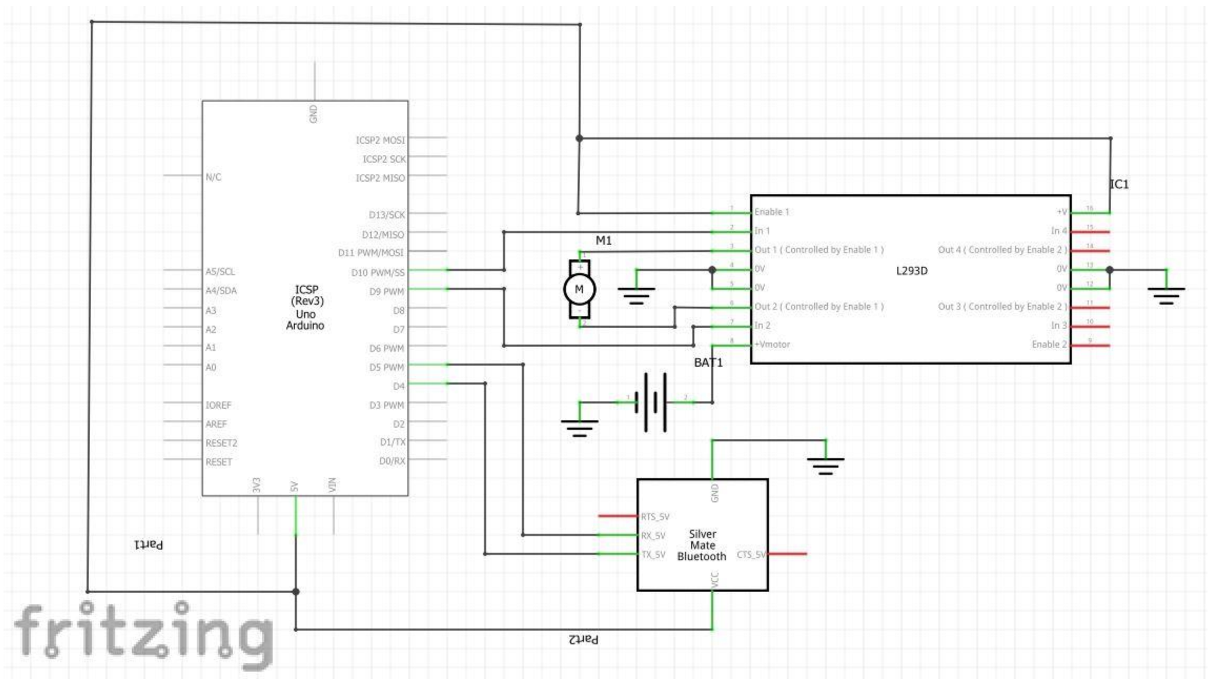
The HC-06 will be receiving serial data when sent out from the master Bluetooth device, which will be communicated to the Arduino through its receive/transmit pins, RX/TX, connected to pins 5 and 4 of the board. Its GND and VCC pins will be respectively connected to GND and +5V.

The H-bridge in the L293D will allow us to control the rotational direction of the motor, by opening or closing a pair of switches, 4 of which are arranged in the shape of an H, hence the name.


H bridge topology
Connecting the load in one direction
Connecting the load in the other direction
VCC
Load
GND
VCC
Load
GND
VCC
Load
GND

This motor driver is also capable of driving two motors at different speeds, but only one will be used for this project.

To enable the motor, connect " Enable 1, 2 " (pin1) and " Vcc1 " (pin 16) to +5V. Connect " Vcc2 " (pin 8) to the positive of the 12V battery pack. Connect pins 4, 5, 12 and 13 to GND. Connect " Input 1 " (pin 2) to pin 10 of the Arduino, which will transmit the pwm2 signal, and " Input 2 " (pin 7) to pin 9, for pwm1 . Switching between these will allow us to switch the rotational direction. Finally, connect the DC motor to outputs 1 and 2 (pins 3 and 6).



Comments have been added to help explain the code, but some more clarification may be needed.

pwm1 and pwm2 are opposites, and allow us to switch the rotational direction. The serial data sent by the app is a string, c , which is where the speed and direction data will be. Speed will depend on the input. The '1' corresponds to slow speed, '2' to medium and '3' to fast speed of the fan.

Please check out the following video of the execution of our project.

https://drive.google.com/file/d/1syS-Zy5WWXjmgIjcJXaLNiNCWUsbD71Q/view?usp=sharing

Applications and Future Scopes

The circuit can be use in the following cases:

- Remote Controlled Cars.
- Smart Fans.
- Smart Bulbs.
- Roller Skates.
- Wirelessly controlled Cars.

Conclusions

This project aimed to design and implement a system for controlling the speed and direction of a DC motor using Bluetooth through a mobile app. The system consisted of an Arduino board, an H-bridge motor driver, and a Bluetooth module. The mobile app was developed using MIT App Inventor.

The PWM (Pulse Width Modulation) technique was used to control the speed of the DC motor. The duty cycle of the PWM signal was varied based on the value received from the mobile app. The direction of the motor was controlled using a separate pin of the H-bridge motor driver.

The system was tested, and the results showed that the speed and direction of the motor could be controlled smoothly using the mobile app. The system can be used in various applications, including robotics, automation, and home appliances.

In conclusion, this project successfully demonstrated the control of a DC motor's speed and direction using Bluetooth through a mobile app. This project can be further improved by adding additional features such as motor current monitoring, motor temperature monitoring, and motor overload protection.

The hardware used for this project included an Arduino Uno board, an L293D H-bridge motor driver, a DC motor, and a HC-05 Bluetooth module. The H-bridge motor driver was used to control the direction of the motor, while the PWM technique was used to control the speed.

The software for this project was developed using the Arduino IDE and MIT App Inventor. The Arduino code included a function to receive data from the Bluetooth module and convert it into a PWM signal to control the speed of the motor. Another function was used to control the direction of the motor using the H-bridge motor driver. The mobile app was developed using MIT App Inventor and included buttons for controlling the speed and direction of the motor.

To test the system, the Arduino board was connected to a power source, and the Bluetooth module was paired with a mobile phone. The mobile app was then used to send commands to the Arduino board to control the speed and direction of the motor.

The project was successful in achieving its objectives, and the system was able to control the speed and direction of the motor through Bluetooth using the mobile app. This project can be used in various applications, such as robotics, automation, and home appliances, to provide a wireless means of controlling the speed and direction of a DC motor.

THANK

YOU