MiniDRIVE

A Teleoperated Mobile Robot

Project Report
(2020)

Authors

Tanmay Vilas Samak

Chinmay Vilas Samak

Table of Contents

1. Introduction 1
2. Mechanical System 2
    2.1 Mechanical System Overview 2
    2.2 Chassis Design 2
    2.3 Differential Drive 3
3. Electrical System 3
    3.1 Electrical System Overview 4
    3.2 Electrical System Layout 4
    3.3 Microcontroller 6
    3.4 Bluetooth Module 7
    3.5 Motor Driver 7
    3.6 Motors 8
4. Robot Software 8
    4.1 Software Overview 8
    4.2 MiniDRIVE Algorithm 9
5. Communication System 9
    5.1 Communication System Overview 9
    5.2 Serial Communication Protocols 10
    5.3 UART 10
        5.3.1 Baud Rate 10
        5.3.2 Transmitter (Tx) 11
        5.3.3 Receiver 11
        5.3.4 Frame Format 11
    5.4 UART in ATmega16 Microcontroller 12
        5.4.1 UART Pins on ATmega16 12
        5.4.2 UART Registers in ATmega16 12
        5.4.3 UART in ATmega16 Workflow 15
    5.5 Bluetooth Link 16
6. Control Software 17
    6.1 Smartphone 17
    6.2 Laptop PC 18
7. Supplemental Materials 18
8. References 18

1. Introduction

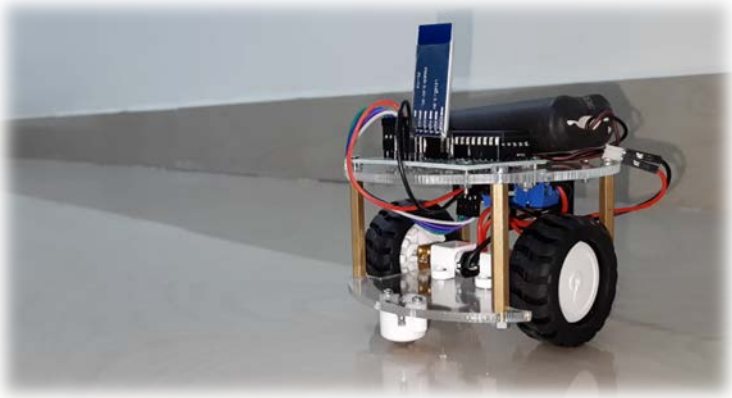


Figure 1. *MiniDRIVE - A Teleoperated Mobile Robot*

MiniDRIVE is a teleoperated (remotely controlled) differentially driven mobile robot which is intended to be controlled by a human or centralized computer and takes no positive action autonomously.

A remotely controlled robot may be defined as a mobile device that is controlled by a means that does not restrict its motion with an origin external to the device. This often includes a radio control device, a cable (tethered connection) between controller and robot or an infrared or Bluetooth link between the controller and robot.

In this project we are using a Bluetooth link to control our robot, which is a wireless technology standard used for exchanging data between fixed and mobile devices over short distances using short-wavelength ultra-high frequency (UHF) radio waves. This project utilizes a Bluetooth based communication interface since it gives a wider range of controllability and better efficiency. It also increases system flexibility by offering an advantage of changing the remote controller anytime, meaning that we can use any computing devices including mobile phones, tablet PCs, laptop PCs, etc. Moreover, physical barriers such as walls, doors, etc. do not effect in controlling the robot (within a moderately short span from the controller device).

The remote controller in this project is any personal computing device (PCD) with Bluetooth (such as smartphones, laptop PCs, etc.) which hosts the control software capable of communicating with the robot over a serial communication port (Bluetooth link) and transmitting the control signals remotely to the robot. In this project, we have used a Bluetooth module in our robot which connects to the PCD's Bluetooth, that allows us to communicate and take command over it.

User(s) can perform various actions like commanding the robot to move forward, backward, left and right using control signals that are sent from the PCD to the robot. Thus, the robot is capable of proceeding accurately to a target area, maneuvering within that area to fulfil its mission and returning equally accurately and safely to base.

The task of controlling the robot is accomplished by the on-board microcontroller – ATmega16. Microcontrollers have played a major role in the domain of mobile robotics and have made it easier to convert simple analog and digital signal to physical movements. Based on the command received from the remote PCD (an ASCII character denoting a particular control signal), the microcontroller generates digital signals for the motor-driver module for direction control of the motors (and pulse width modulated digital signals for speed control, if required) which causes the robot to maneuver in a specific direction owing to the differential drive mechanism.

2. Mechanical System

The mechanical system of the robot functions to provide a rigid support and mounting provision for all of its functional components.

2.1 Mechanical System Overview

The mechanical sub-system of the robot consists of:

- Chassis
- Mounting Provision for Electrical Components
- Mounting Brackets for Motors
- Wheels

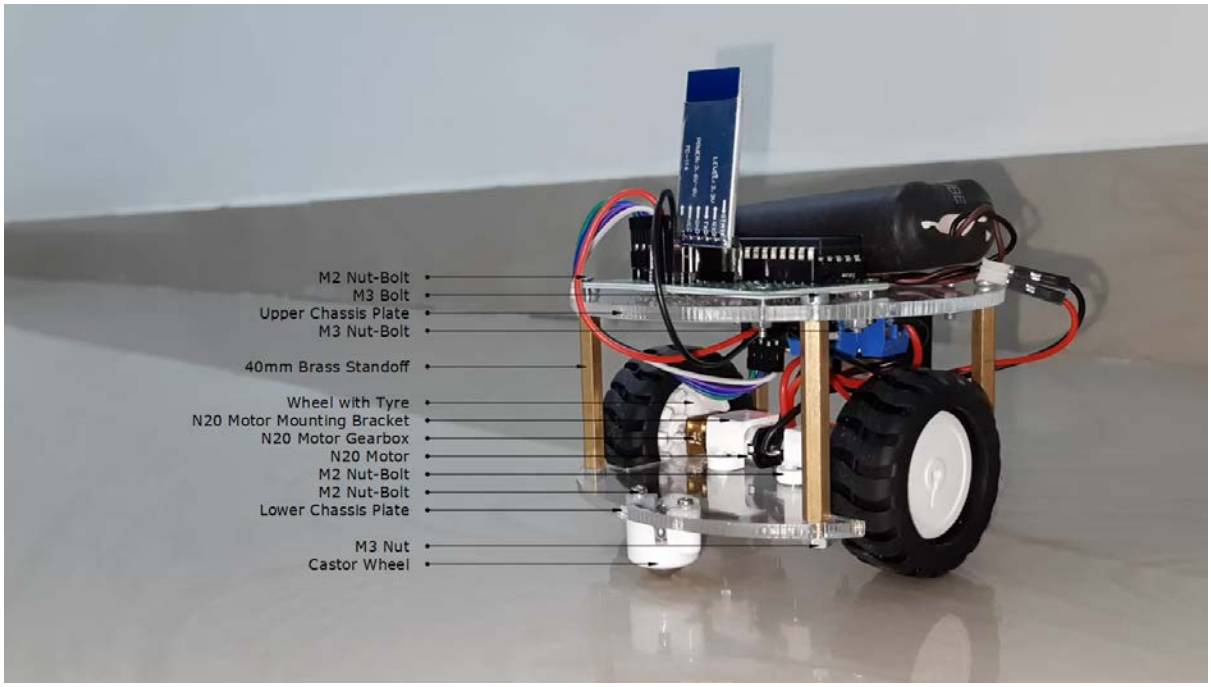


Figure 2. *MiniDRIVE - Mechanical System Overview*

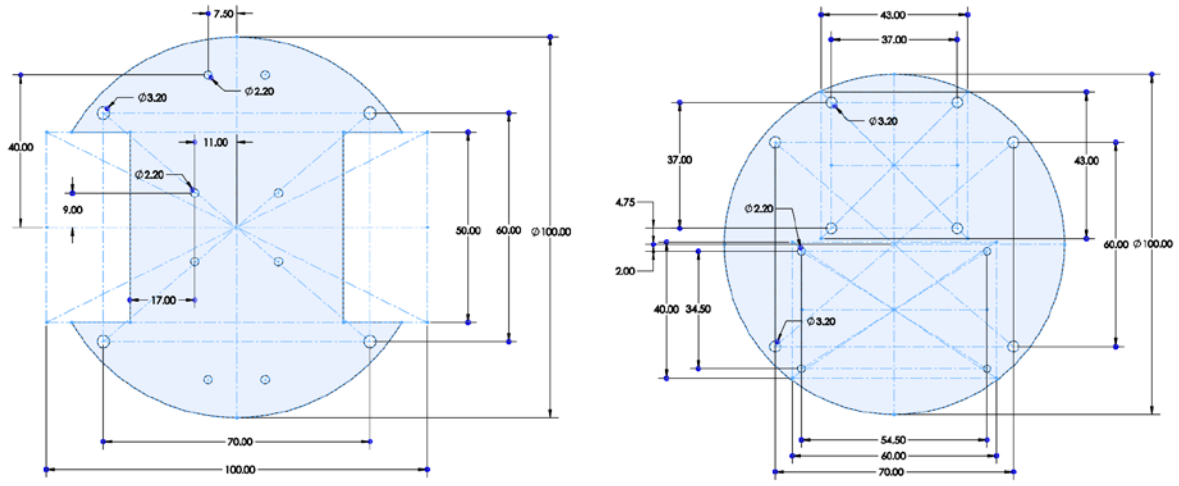
The chassis consists of two acrylic plates separated by 40 mm using standoff spacers. The lower plate has mounting provisions for two caster wheels and mounting brackets for both the motors whereas the upper plate has mounting provisions for the main circuit board and the motor driver, and also has provisional space for accommodating the battery pack. M2 and M3 metric nut-bolts with diametric dimensions of 2 mm and 3 mm respectively are used to fasten the said components firmly to the chassis of the robot.

2.2 Chassis Design

The chassis of the robot is segmented in two parts. It consists of the following two plates connected using four M3 X 40 mm M-F Brass Hex Threaded Pillar Standoff Spacers and standard M3 Nut-Bolts.

- Lower Chassis Plate
- Upper Chassis Plate

The chassis plates were designed considering all the required mounting provisions and their dimensions using SolidWorks CAE tool. The designs were then exported as DXF files to be imported in CorelDraw for the purpose of manufacturing.



(a) (b)

Figure 3. *MiniDRIVE - Chassis Design (a) Upper Chassis Plate and (b) Lower Chassis Plate*

The material for the plates was chosen to be transparent acrylic and the plates were manufactured by laser-cutting 3mm thick sheets of the said material using the CorelDraw graphic files developed earlier.

2.3 Differential Drive

A differentially driven (wheeled or steered) robot is a mobile robot whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels and hence does not require an additional steering motion. To balance the robot, additional wheels or casters may be added.

If both the wheels are driven in the same direction and speed, the robot will go in a straight line. If both wheels are turned with equal speed in opposite directions, as is clear from the diagram shown, the robot will rotate about the central point of the axis. Otherwise, depending on the speed of rotation and its direction, the center of rotation may fall anywhere on the line defined by the two contact points of the wheels. While the robot is traveling in a straight line, the center of rotation is an infinite distance from the robot. Since the direction of the robot is dependent on the rate and direction of rotation of the two driven wheels, these quantities should be sensed and controlled precisely.

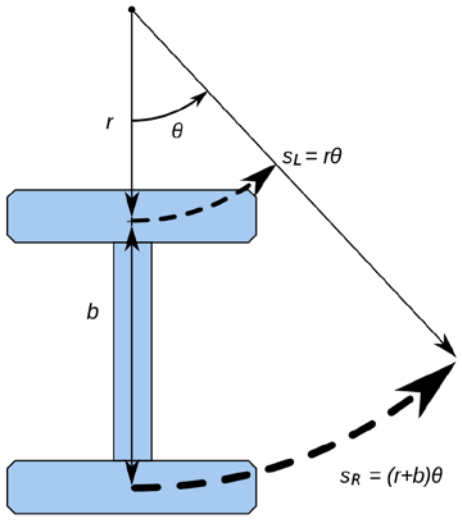


Figure 4. *Differential Drive Mechanics*

A differentially steered robot is similar to the differential gears used in automobiles in that both the wheels can have different rates of rotations, but unlike the differential gearing system, a differentially steered system will have both the wheels powered. Differentially driven wheels is used extensively in mobile robotics, since their motion is easy to program and can be well controlled. Virtually all consumer robots on the market today use differential drive primarily for its low cost and simplicity.

3. Electrical System

The electrical system of the robot functions to route and establish all the required electrical connections connecting various electrical components that carry out the power distribution and signaling operations.

3.1 Electrical System Overview

The electrical sub-system of the robot consists of:

- Battery Pack – 8 V DC Li-Ion Battery Pack
- Microcontroller – Atmega16
- Bluetooth Module – HC-05
- Motor Driver – L298N
- Motors – N20 Micro-Gear 12 V 100 RPM DC Motors

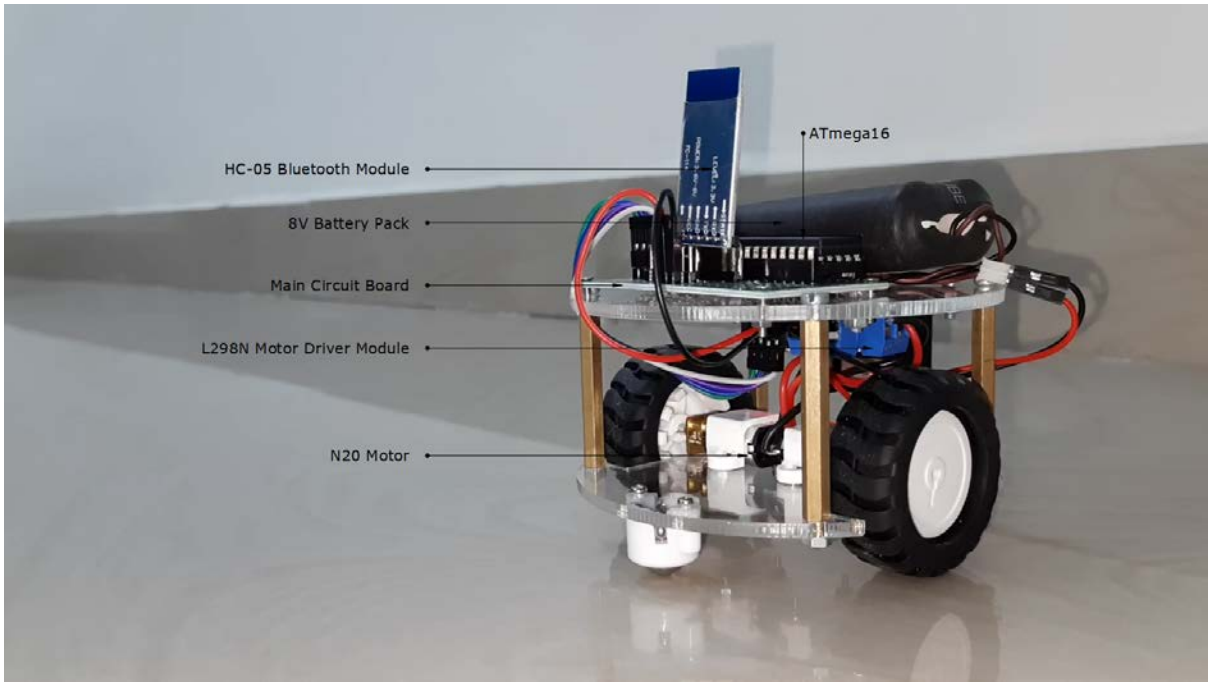


Figure 5. *MiniDRIVE - Electrical System Overview*

3.2 Electrical System Layout

The entire electrical system of MiniDRIVE can be segregated in two sub-systems:

- Power Distribution System
- Signaling and Communication System

Power Distribution System

The high voltage (8 V DC) from the battery is fed to the motor driver and the converted logic level voltage (5 V DC) obtained from it is fed to all the digital components of the robot viz. microcontroller and Bluetooth module. The motors receive direct supply (8 V DC) from the battery when the motor driver is commanded to operate the specific motor(s) by the microcontroller.

Signaling and Communication System

The HC-05 Bluetooth module receives commands from remote PCD and transfers them to the interfaced microcontroller via UART (PORTD Pin 0 – PD0). Based on the commands received from the PCD, the microcontroller then generates control signals for the motor driver using four pins of PORTB (PB0 – PB3) (two for each motor) which then operates the required motor(s) for the robot to maneuver in a specific direction.

The entire electrical system layout of the robot displaying the electrical routing of all the components is shown in the following figure.

Note that the red and black connections in the following figure are a part of the power distribution system (red indicating positive and black denoting negative terminals of the on-board DC power supply) whereas the yellow and green connections are a part of the signaling and communication system.

Also note that the female header strip (first from the left) with wire codes orange to black is a provision for in-circuit serial programming the ATmega16 via USBASP.

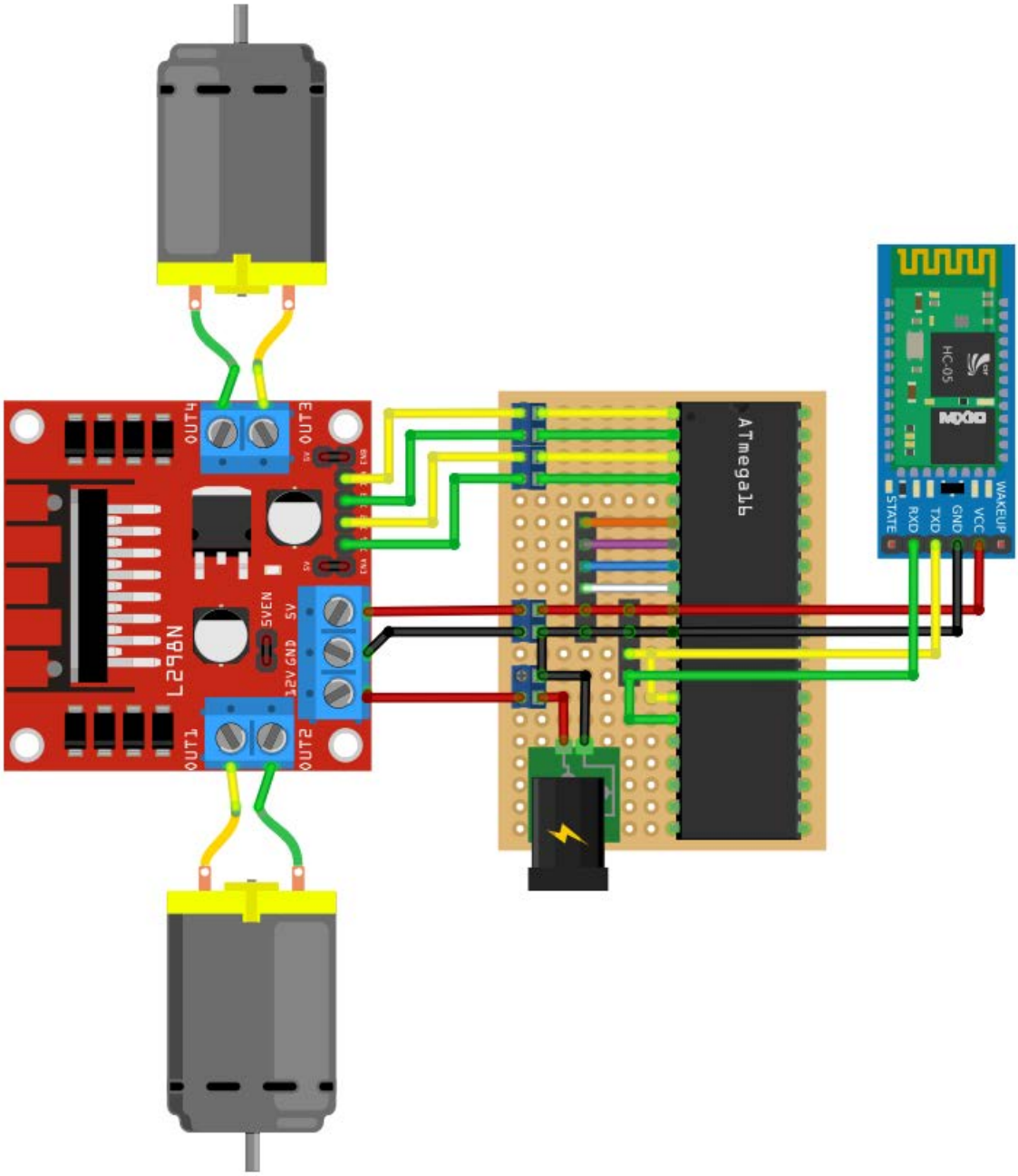
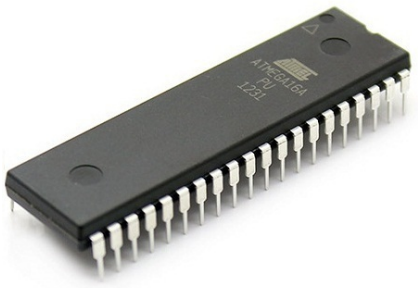


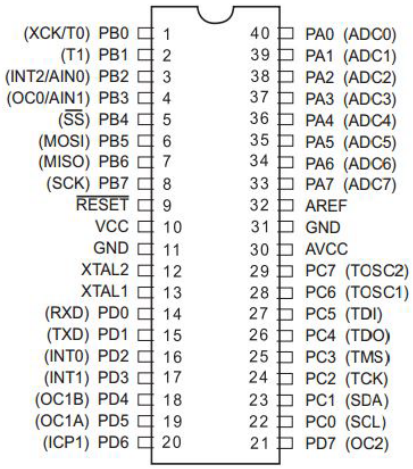
Figure 6. *MiniDRIVE Electrical System Layout*

3.3 Microcontroller

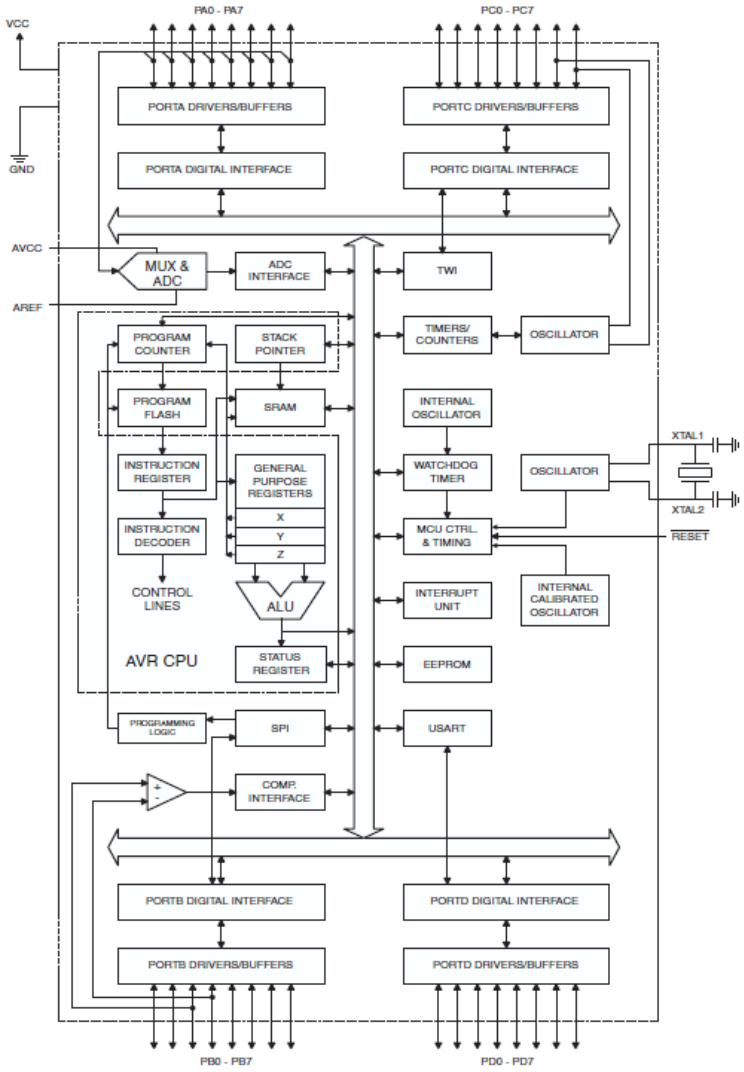
Atmega16 is an 8-bit Atmel® AVR® microcontroller based on advanced RISC (Reduced Instruction Set Computing) architecture. It is a 40-pin low power microcontroller which is developed using CMOS technology.



(a)



(b)



(c)

Figure 7. *ATmega16 (a) DIP IC, (b) Pin Diagram and (c) Architecture*

ATmega16 is a single chip microcontroller that comes with CPU, ROM, RAM, EEPROM, Timers, Counters, ADC, Analog Comparator, USART, SPI and four 8-bit ports namely PORTA, PORTB, PORTC and PORTD where each port consists of 8 I/O pins.

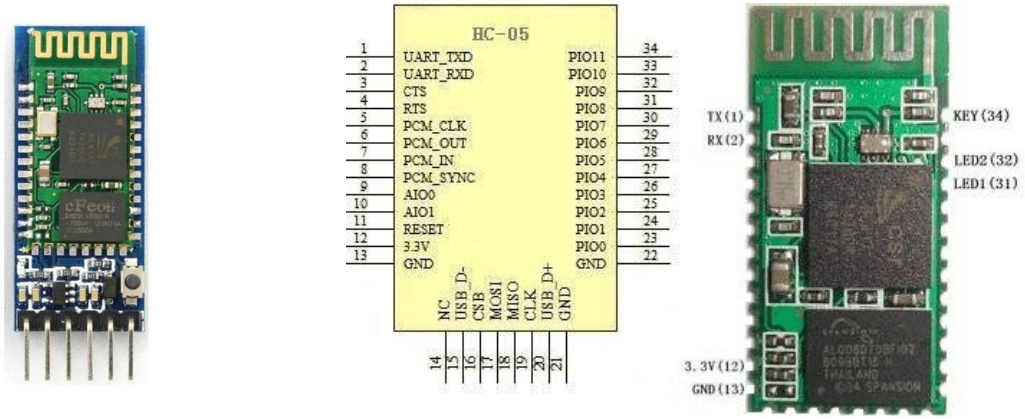
Atmega16 has built-in registers that are used to make a connection between CPU and external peripherals devices. CPU has no direct connection with external devices. It can take input by reading registers and give output by writing registers.

Atmega16 has 32 x 8 built-in general purpose working registers that are used to make a connection between CPU and external peripherals devices. CPU has no direct connection with external devices. It can take input by reading certain registers and give output by writing to certain registers.

Atmega16 comes with two 8-bit timers and one 16-bit timer. All these timers can be used as counters when they are optimized to count the external signal. It comes with 1KB of static RAM (SRAM) which is a volatile memory i.e. stores information for short period of time and highly depends on the constant power supply. Another 16KB of flash memory, also known as ROM, is also incorporated in the device which is non-volatile in nature and can store information for long period of time and doesn't lose any information when the power supply is disconnected. Additionally, it also has a dedicated 512 Bytes of EEPROM. Atmega16 works on a maximum frequency of 16MHz where instructions are executed in one machine cycle.

3.4 Bluetooth Module

HC-05 module is an easy to use low-power Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.



(a) (b)

Figure 8. *HC-05 Bluetooth Module (a) Breakout Board and (b) Pin Diagram*

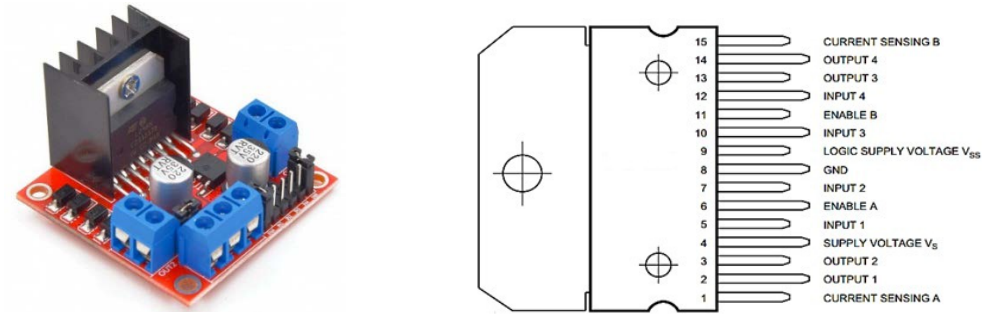
This serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3 Mbps modulation with complete 2.4 GHz radio transceiver and baseband. It uses CSR Bluecore 04 external single chip Bluetooth system with CMOS technology with AFH (Adaptive Frequency Hopping) feature.

It can operate in two dedicated modes:

- **Pairing Mode** – This mode is used to connect and communicate with a host device.
- **Command Mode** – This mode is used to configure the module using AT commands.

3.5 Motor Driver

L298N is a high voltage, high current dual full-bridge driver designed to accept standard TTL levels and drive inductive loads such as relays, solenoids, DC motors and stepper motors.



(a) (b)

Figure 9. *L298N Motor Driver (a) Breakout Board and (b) Pin Diagram*

Two enable inputs are provided to enable or disable the device independent of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage while the motors can draw enough power.

3.6 Motors

The N20 Micro Gear 12 V 100 RPM DC Motor is lightweight, high torque, and low speed brushed DC motor. It is equipped with a gearbox assembly so as to increase the output torque thereby exhibiting excellent stall characteristics.

Figure 10. *N20 12 V 100 RPM Micro-Gear DC Motor*

The N20 Micro Gear 12 V 100 RPM DC Motor has a cross section of 10 mm x 12 mm and can therefore fit in complex spaces for small-scale applications.

The D-shaped gearbox output shaft is 9 mm long and 3 mm in diameter which makes it easy to mount and lock a wheel.

4. Robot Software

4.1 Software Overview

The software for MiniDRIVE was written in Embedded C language including 2 header files and consisting 3 macro and function definitions each.

The header files used were:

- *avr/io.h* – AVR device-specific IO definitions (for ATmega16 microcontroller)
- *util/delay.h* – Convenience functions for busy-wait delay loops (for time delays)

The macro definitions were made in order to define the CPU frequency, baud rate for serial communication and the converted baud rate value to be stored in USART Baud Rate Register (UBRR) corresponding to asynchronous normal mode of operation.

The three functions defined in the program were:

- *UART_Initialize()* – This function initializes UART with the defined communication parameters.
- *UART_Receive()* – This function enables reception of serial data packets via UART and returns the received 8-bit data packet.
- *main()* – This is the main program that first calls the necessary functions and then runs the entire algorithm in an infinite loop.

The program was burnt onto ATmega16 microcontroller using USB AVR Serial Programmer (USBASP). ATmega16 pins 6-11 were used for programming the microcontroller via USBASP.

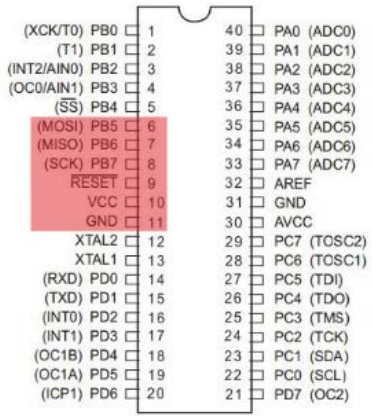


Figure 11. *ATmega16 USBASP Programming Pins*

4.2 MiniDRIVE Algorithm

```
ALGORITHM
1:   PORTB ← 0b11111111;                  [declare the output port]
2:   UART_Initialize();                   [start the serial communication]
3:   _delay_ms(50);                       [delay for stable communication]
4:   while(1) {                           [infinite loop]
5:       dataPacket ← UART_Recieve();     [store the received byte in a variable dataPacket]
6:       if (dataPacket == 'w') {         [motor control case structure based on the byte received]
             move forward
         }
         if (dataPacket == 's') {
             move backward
         }
         if (dataPacket == 'a') {
             turn left
         }
         if (dataPacket == 'd') {
             turn right
         }
         else {
             stop
         }
     }
```

5. Communication System

The communication system of the robot functions to establish an uninterrupted wireless connection link between the robot and the remote PCD in order to effectively control the robot motion.

5.1 Communication System Overview

Communication is an exchange of information (data) between two or more entities over a medium. Communication systems can be categorized based upon:

- Based on Mode of Operation
  - Simplex – One-way communication (Master → Slave)
  - Half-Duplex – Two-way communication but not at the same time (Master ↔ Slave)
  - Full-Duplex – Full two-way communication at the same time (Master ↔ Slave)
- Based on Method of Transmission
  - Serial Communication – Data sent or received one bit at a time over a single channel
  - Parallel Communication - Data bits sent or received at a time over multiple channels

In this project, we have adopted full-duplex serial communication scheme.

5.2 Serial Communication Protocols

A communication protocol is a set of rules or standards that the transmitter and receiver must agree to for smooth and uninterrupted communication.

- Physical Interface
- Rate of Transmission
- Synchronization
- Frame Format
- Error Detection

There are two types of serial communication protocols:

- Synchronous Protocols
  - Data bits are synchronized with a clock
  - Transmitter and receiver operate on the same clock (transmitter supplies the clock)
  - Transmitter is the master and receiver is the slave device
- Asynchronous Protocols
  - No clock, hence asynchronous
  - Start and stop bits are required for synchronization
  - Start bit is LOW
  - Stop bit is HIGH

Table 1. *Comparison of Synchronous and Asynchronous Communication Protocols*

Synchronous Protocols	Asynchronous Protocols
Requires clock for synchronization	Requires start and stop bits for synchronization since there is no clock
Higher data rates can be achieved since there are no start and stop bits	Lower data rates can be achieved since start and stop bits are present for synchronization
Complex setup and more hardware involved	Simple setup with less hardware involved
Expensive	Cheap

In this project, we have adopted asynchronous serial communication protocol.

5.3 UART

UART is an acronym for Universal Asynchronous Receiver-Transmitter, which is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable.

- **Universal:** Baud rate and frame format are configurable
- **Asynchronous:** There is no clock, start and stop bits are used for synchronization
- **Receiver-Transmitter:** Devices communicating using asynchronous communication protocol

5.3.1 Baud Rate

Baud is a unit of transmission speed equal to the number of times a signal changes state per second. For signals with only two possible states (i.e. digital signals) one baud is equal to one bit per second.

Transmitter and receiver must have an exact same baud rate for smooth and effective communication. Some of the standard baud rate settings include 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, etc.

In this project, we have adopted 4800 baud data transmission rate.

5.3.2 Transmitter (Tx)

Transmitter (Tx) is the line from which data is sent out to the receiver. It should be connected to the receiver (Rx) of the device with which it is communicating.

5.3.3 Receiver

Receiver (Rx) is the line from which data is received from the transmitter. It should be connected to the transmitter (Tx) of the device with which it is communicating.

5.3.4 Frame Format

A frame refers to the entire data packet which is being transmitted or received during a communication cycle. Each communication has its own frame format.

Table 2. *UART Frame Format*

Bit Name	Number of Bits (Size)	Bit Value (Logic State)
Start Bit	1	LOW (0)
Data Bits	5-9	LOW/HIGH (1/0)
Parity Bit	0-1	LOW/HIGH (1/0)
Stop Bit	1-2	HIGH (1)

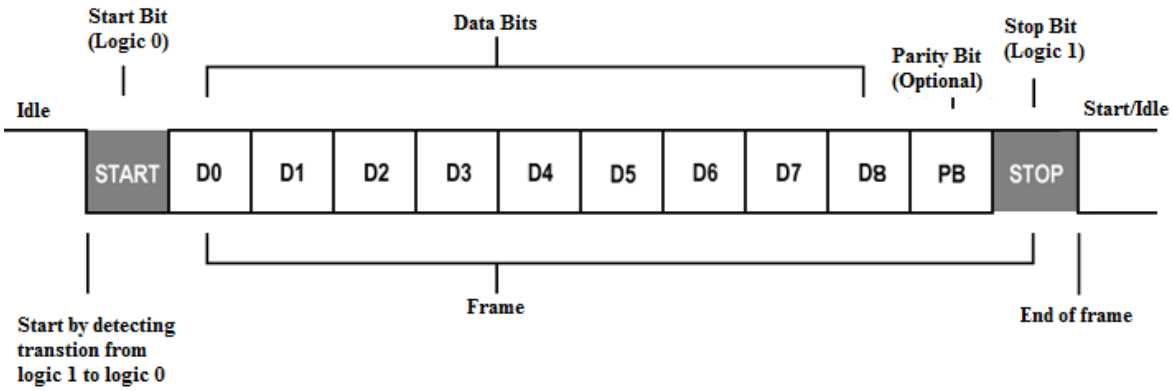


Figure 12. *UART Frame Format*

The **start bit** indicates the start of communication or the start of a new frame. There is only one start bit and it is always LOW (0).

The **data bits** are binary numerals of data being sent by the transmitter. There can be a minimum of 5 and a maximum of 9 data bits in a single data frame. The least significant bit (LSB) of the data packet is always transmitted first.

The **parity bit** is inserted after the last data bit and before the first stop bit. It is used for error detection and is optional. There are two types of parity:

- Even Parity: The number of 1s present in data bits including the parity bit is even
- Odd Parity: The number of 1s present in data bits including the parity bit is odd

The **stop bit** indicates end of a frame. It is always HIGH (1). A frame needs to have a minimum of 1 stop bit compulsorily. The second stop bit is optional.

If a start bit is not sent following a stop bit, the communication line is said to be IDLE (HIGH).

5.4 UART in ATmega16 Microcontroller

The features of UART (USART) in ATmega16 microcontroller (as per the datasheet) are:

- Full-duplex operation (independent serial receive and transmit registers)
- Asynchronous or synchronous operation
- Master or slave clocked synchronous operation
- High resolution baud rate generator
- Supports serial frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or even parity generation and parity check supported by hardware
- Data over-run detection
- Framing error detection
- Noise filtering includes false start bit detection and digital low pass filter
- Three separate interrupts on TX Complete, TX Data Register Empty, and RX Complete
- Multi-processor communication mode
- Double speed asynchronous communication mode

5.4.1 UART Pins on ATmega16

The USART of ATmega16 consists of three pins:

1. **XCK** – USART Clock Pin (Pin 1)
2. **RXD** – USART Receiver Pin (Pin 14)
3. **TXD** – USART Transmitter Pin (Pin 15)

However, since this project adopted UART (asynchronous mode of communication), the XCK pin was not utilized.

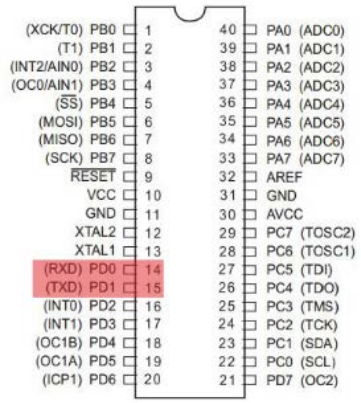


Figure 13. ATmega16 UART Pins

5.4.2 UART Registers in ATmega16

ATmega16 has 3 set of registers to handle operations related to UART (USART).

1. USART Baud Rate Register (UBRR)
    - UBRRH (higher byte)
    - UBRRL (lower byte)
2. USART Control and Status Register (UCSR)
    - UCSRA
    - UCSRB
    - UCSRC
3. USART Data Register (UDR)

5.4.2.1 USART Baud Rate Register (UBRR)

UBRR is a 16-bit register used for setting baud rate for serial communication.

Figure 14. *ATmega16 - USART Baud Rate Register (UBRRH and UBRRL)*

Bit 15 – URSEL: USART Register Select
This bit selects between accessing **UCSRC** or **UBRRH**, since both registers share the same address. The URSEL must be one when writing to UCSRC or else data will be written in UBRRH.

Bit 11:0 – UBRR11:0: USART Baud Rate Register

These bits are used to define baud rate based on the CPU frequency.

5.4.2.2 USART Control and Status Register (UCSR)

UCSRA: USART Control and Status Register A

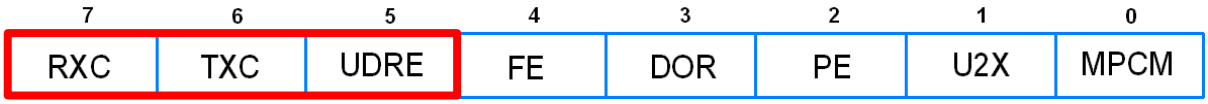


Figure 15. *ATmega16 - USART Control and Status Register A (UCSRA)*

Bit 7 – RXC: USART Receive Complete

This flag bit is set when there is unread data in UDR. The RXC flag can be used to generate a Receive Complete interrupt.

Bit 6 – TXC: USART Transmit Complete

This flag bit is set when the entire frame from transmit buffer is shifted out and there is no new data currently present in the transmit buffer (UDR). The TXC flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a logical HIGH to its bit location. The TXC Flag can generate a Transmit Complete interrupt.

Bit 5 – UDRE: USART Data Register Empty

If UDRE bit is HIGH, the buffer is empty which indicates the transmit buffer (UDR) is ready to receive new data. The UDRE flag can generate a Data Register Empty interrupt. UDRE is set after a reset to indicate that the transmitter is ready.

Bit 4 – FE: Frame Error

Bit 3 – DOR: Data OverRun

This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters) and a new character is waiting in the receive Shift Register.

Bit 2 – PE: Parity Error

Bit 1 – U2X: Double the USART Transmission Speed

Bit 0 – MPCM: Multi-Processor Communication Mode

UCSRB: USART Control and Status Register B

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

Figure 16. ATmega16 - USART Control and Status Register B (UCSRB)

Bit 7 – RXCIE: RX Complete Interrupt Enable

Writing one to this bit enables interrupt on the RXC flag.

Bit 6 – TXCIE: TX Complete Interrupt Enable

Writing one to this bit enables interrupt on the TXC flag.

Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable

Writing one to this bit enables interrupt on the UDRE flag.

Bit 4 – RXEN: Receiver Enable

Writing one to this bit enables the USART Receiver.

Bit 3 – TXEN: Transmitter Enable

Writing one to this bit enables the USART Transmitter.

Bit 2 – UCSZ2: Character Size

The UCSZ2 bit combined with the UCSZ1:0 bits in UCSRC set the number of data bits (Character Size) in a frame that receiver and transmitter use.

Bit 1 – RXB8: Receive Data Bit 8

Bit 0 – TXB8: Transmit Data Bit 8

UCSRC: USART Control and Status Register C

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	USCZ1	USCZ0	UCPOL

Figure 17. ATmega16 - USART Control and Status Register C (UCSRC)

Bit 7 – URSEL: Register Select

This bit selects between accessing UCSRC or UBRRH, since both registers share the same address. The URSEL must be one when writing to UCSRC or else data will be written in UBRRH.

Bit 6 – UMSEL: USART Mode Select

This bit selects between asynchronous and synchronous mode of operation.

- **0** = Asynchronous Operation
- **1** = Synchronous Operation

Bit 5:4 – UPM1:0: Parity Mode

These bits enable and set type of parity generation and check. If a mismatch is detected, the PE flag in UCSRA will be set.

Table 3. Parity Mode

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Bit 3 – USBS: Stop Bit Select

This bit selects the number of Stop Bits to be inserted by the Transmitter. The Receiver ignores this setting.

- **0** = 1-bit
- **1** = 2-bit

Bit 2:1 – UCSZ1:0: Character Size

The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB set the number of data bits (Character Size) in a frame that receiver and transmitter use.

Bit 0 – UCPOL: Clock Polarity

This bit is used for synchronous mode only. One should clear this bit to zero when asynchronous mode is to be used.

5.4.2.3 USART Data Register (UDR)

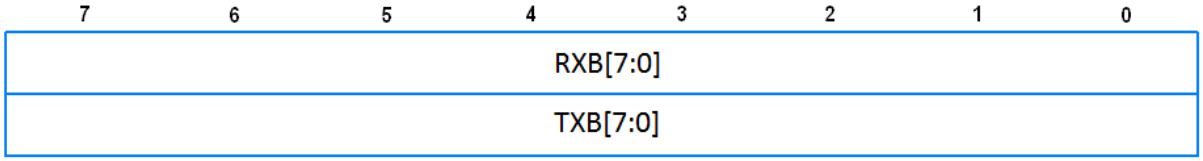


Figure 18. ATmega16 - USART Data Register (UDR)

UDR fundamentally constitutes of two registers, one is Transmitter Byte and other is Receiver Byte. Both share the same register (UDR). When we write to UDR, Tx buffer gets written and when we read from UDR, Rx buffer gets read. Both the buffers use FIFO shift register to transmit the data.

5.4.3 UART in ATmega16 Workflow

5.4.3.1 Set Baud Rate

As stated earlier, baud rate can be set using the USART Baud Rate Register (UBRRH and UBRRL). As per the datasheet of ATmega16 microcontroller, the converted baud rate value to be stored in USART Baud Rate Register (UBRR) corresponding to asynchronous normal mode of operation can be calculated as follows.

$$UBRR = \frac{F_{CPU}}{Baud\ Rate * 16} - 1$$

In this project the data transmission was chosen to be 4800 baud (bps) as stated earlier while the CPU frequency was set to 1 MHz. Therefore, the BURR value was calculated to be:

$$UBRR = \frac{10^6}{4800 * 16} - 1 = \frac{10^6}{76800} - 1 = 13.02 - 1 = 12.02 \cong 12$$

5.4.3.2 Enable Transmitter and Receiver

In order to enable transmitter and receiver in ATmega16 microcontroller, TXEN and RXEN bits of UCSRB need to be set. However, since this project did not include any data transmission from the microcontroller side, only receiver was enabled.

5.4.3.3 Select Frame Format

The UCSZ2 bit in UCSRB combined with the UCSZ1:0 bits in UCSRC set the number of data bits (Character Size) in a frame that receiver and transmitter use.

Table 4. *Character Size in ATmega16*

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	**1**	**1**	**8-bit (Default)**
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

5.4.3.4 Check for Complete Reception of Data

7th bit (flag) of UCSRA (RXC – USART Receive Complete) is set when there is unread data in UDR. The RXC flag can be used to generate a Receive Complete interrupt. In other words, when a complete data packet is received in the Rx buffer, RXC flag is set HIGH. Thus, it is necessary to wait until the RXC flag is reset (LOW) so that the whole data packet stored in UDR can be accessed as and when required.

5.4.3.5 Retrieve the Received Data

All the received data is stored in the Rx byte of UDR (i.e. the Receiver Data Buffer) which uses FIFO shift register to transmit the received data byte.

5.5 Bluetooth Link

Bluetooth wireless technology is a short-range radio technology, which is developed for Personal Area Networks (PAN). Bluetooth is a standard developed by a group of electronics manufacturers that allows any sort of electronic equipment, from computers and cell phones to keyboards and headphones, to make its own connections, without wires, cables or any direct action from a user. It is an ad hoc type network operable over a small area such as a room. Bluetooth wireless technology makes it possible to transmit signals over short distances between telephones, computers and other devices and thereby simplifies communication and synchronization between devices. It is a global standard that eliminates wires and cables between both stationary and mobile devices and facilitates both data and voice communication. Bluetooth offers the possibility of ad hoc networks and delivers the ultimate synchronicity between all personal devices. Bluetooth is a dynamic standard where devices can automatically find each other, establish connections, and discover what they can do for each other on an ad hoc basis.

Bluetooth is intended to be a standard that works at two levels:

1. It provides agreement at the physical level - Bluetooth is a radio-frequency standard.
2. It also provides agreement at the next level up, where devices have to agree on when bits are sent, how many will be sent at a time and how the parties in a conversation can be sure that the message received is the same as the message sent.

It was conceived initially by Ericsson, before being adopted by a myriad of other companies. Bluetooth is a standard for a small, cheap radio chip to be plugged into computers, printers, mobile phones, etc. A Bluetooth chip is designed to replace cables by taking the information normally carried by the cable, and transmitting it at a special frequency to a receiver Bluetooth chip, which will then give the information received to the computer, phone, or a something like robot (in our case).

Bluetooth networks (commonly referred to as Piconets) use a master/slave model to control when and where devices can send data. In this model, a single master device can be connected to up to seven different slave devices as shown in the following figure. Any slave device in the Piconet can only be connected to a single master. The master coordinates communication throughout the Piconet. It can send data to any of its slaves and request data from them as well. Slaves are only allowed to transmit to and receive from their master. They can't talk to other slaves in the Piconet.

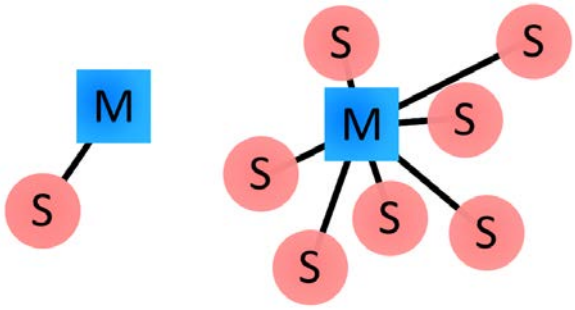


Figure 19. *Bluetooth Master-Slave Piconet Topologies*

6. Control Software

6.1 Smartphone

A smartphone running Android OS was utilized as a remote controller for MiniDRIVE. For this purpose, the following two Android applications were developed:

- BT Terminal
- BT Robot Controller

In **BT Terminal** application, the user can connect to the robot and once connected successfully, can transmit ASCII characters ***w***, ***s***, ***a*** and ***d*** in order to make the robot go forward, backward, turn left and right respectively, and any other ASCII character (e.g. ***x***) to make the robot stop.

On the other hand, the **BT Robot Controller** application is a much powerful one, allowing the user to control the robot in the following 3 possible ways:

1. Control Using Buttons
2. Control Using Voice Commands
3. Control Using Device Orientation

The ASCII character(s) to be transmitted to the robot can as well be configured from within the application itself thereby giving full flexibility to the user.

6.2 Laptop PC

A laptop PC running Windows OS was utilized as a remote controller for MiniDRIVE. For this purpose, the following two applications were utilized:

- PuTTY
- MiniDRIVE

The former is an open source software (available at putty.org) while the later one is developed by us using Processing IDE.

PuTTY is an SSH and telnet client for Windows platform, which allows serial communication to a specific communication port at a specific baud rate. This software was used similar to BT Terminal on Android platform to control the robot by transmitting ASCII characters.

On the other hand, **MiniDRIVE GUI**, an application that we developed, allows the user(s) to control the robot conveniently using the arrow keys on the keyboard, which cannot be achieved using the presently available applications.

7. Supplemental Materials

Following is the list of supplemental materials for this project:

- GitHub Repository:
  https://github.com/Samak-Twins/MiniDRIVE
- Demonstration Video (YouTube):
  https://www.youtube.com/watch?v=Rbxxlpjfh9E
- BT Terminal Android Application:
  https://play.google.com/store/apps/details?id=appinventor.ai_samakbrothers.BT_Terminal
- BT Robot Controller Android Application:
  https://play.google.com/store/apps/details?id=appinventor.ai_samakbrothers.DriveBot_Controller

8. References

1. Atmel® AVR® *ATmega16 – Data Sheet*, Atmel Corporation, San Jose, California, United States, (Accessed 2020).
2. *HC-05 Bluetooth to Serial Port Module – Data Sheet*, ITead Intelligent Systems Co. Ltd., Shenzhen, China, (Accessed 2020).
3. *L298 Dual Full-Bridge Driver – Data Sheet*, STMicroelectronics, Geneva, Switzerland, (Accessed 2020).
4. *G12-N20 Geared Mini DC Motor – Data Sheet*, HandsOn Technology Enterprise, Johor, Malaysia, (Accessed 2020).