Wireless driving shield

Team members:.

Saurabh Pranjale.
Dhruv Patel.
Mansi Sampat
Milie Bhanani

Mentors

Sushen Kashyap
Divya Poddar

Index

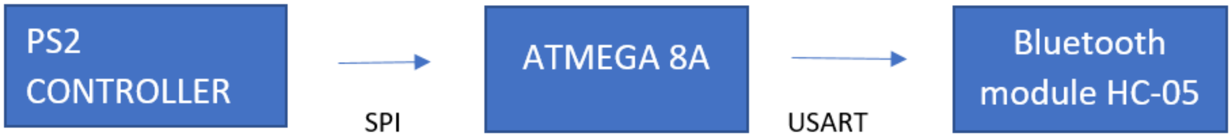
Project abstract

The purpose of the project was to prepare a universal kit which can be used to control any robot wirelessly just by connecting the motors to it. Wireless communication could be through SPI (Serial Peripheral interface), USART, Wi-Fi or RF module.

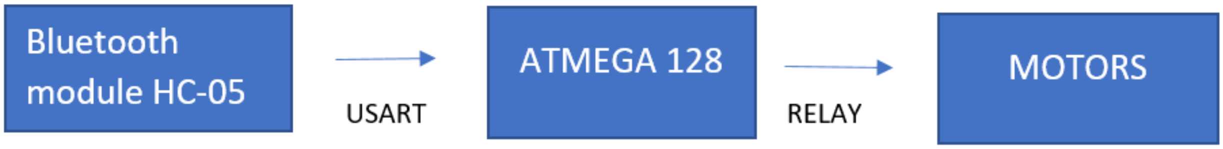
Motivation

For wireless communication, the options available were USART, PS2 , Wi-Fi and RF module. RF module was not used since there may be connectivity issue due to frequency matching.
The project was divided into two modules:

1. PS2:



2. USART:

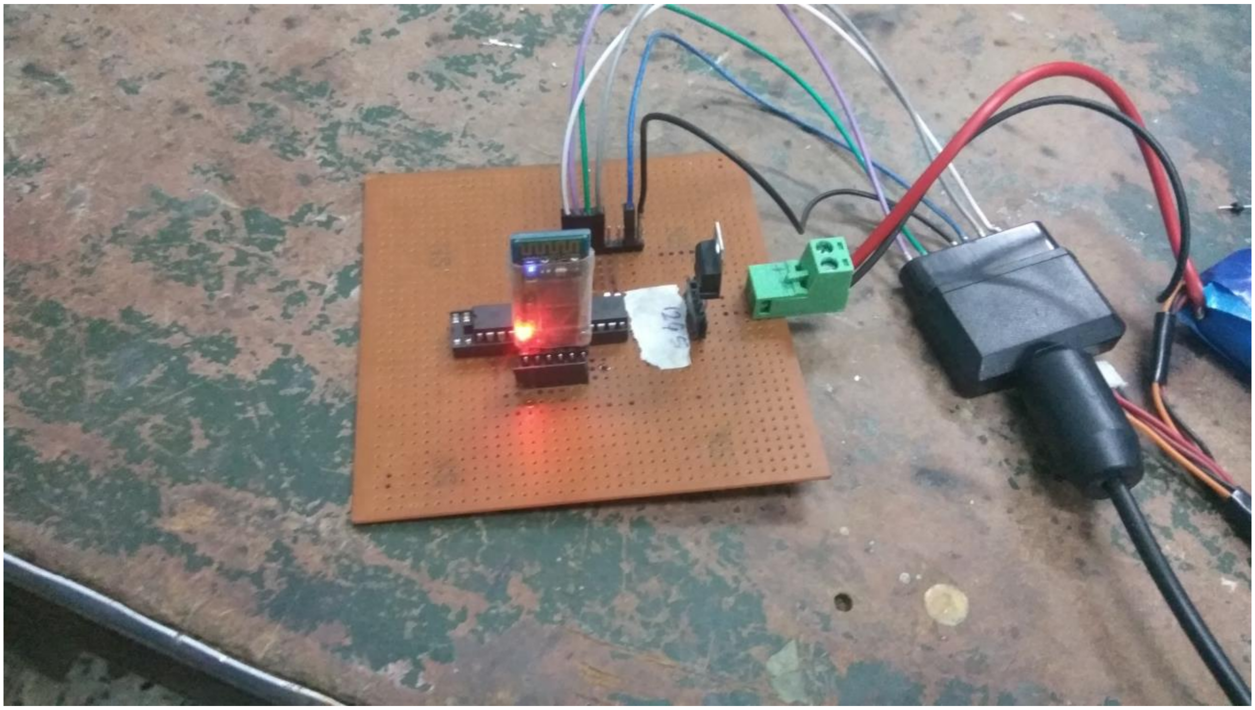


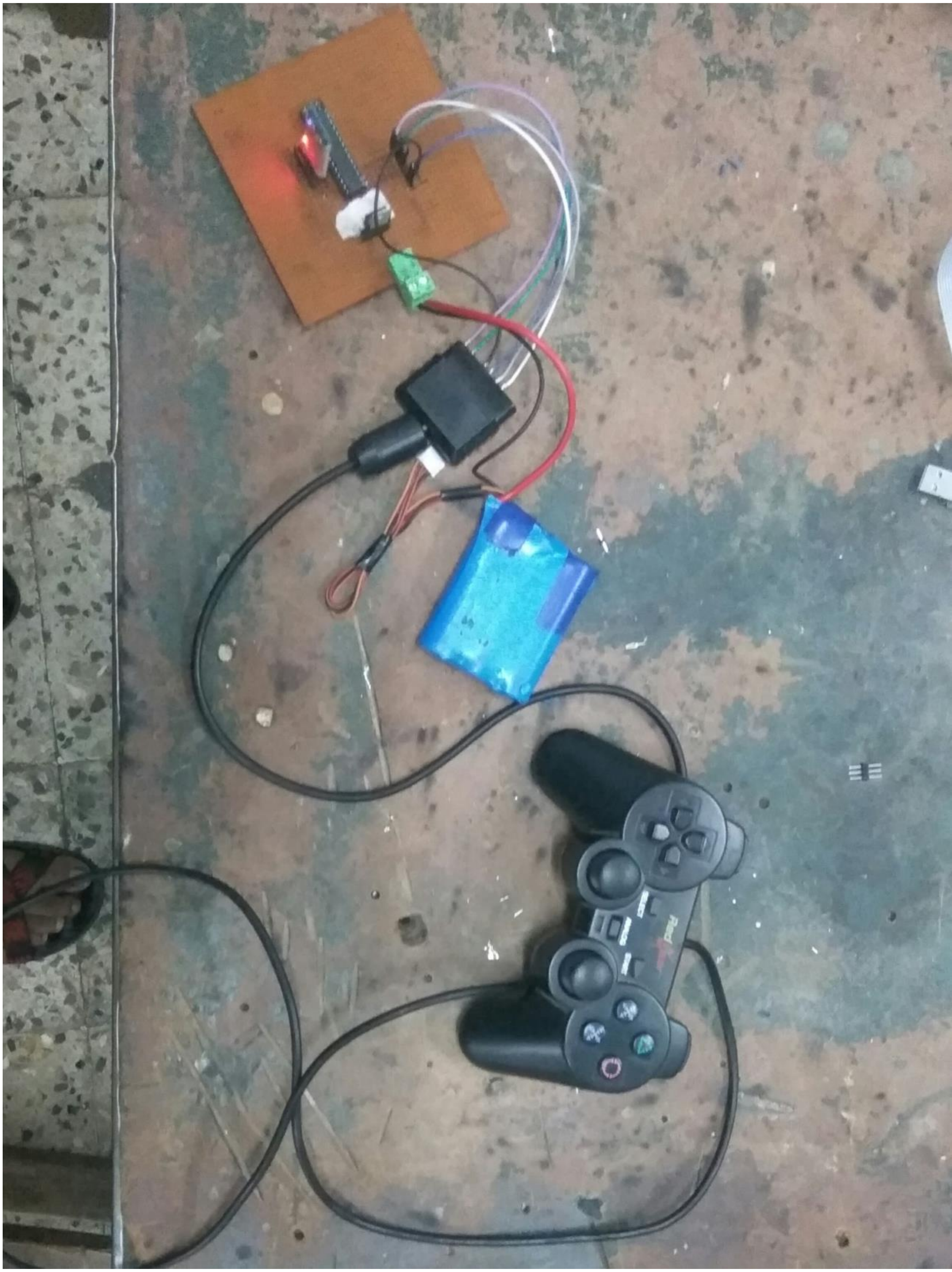
Ps2 controller is connected to microcontroller Atmega8A and data is transmitted through SPI. GND of PS2 with the atmega.Now atmega8a is connected with HC-05 Bluetooth module through UART (COMMUNICATION PROTOCOL).On the other side ATMEGA 128 is connected with HC 05 with same UART Protocol and motor is connected through relay (for switching)and ULN 2003A IC .

WORKING

1. PS2 to atmega8:
    a. Data is sent from ps2 to atmega 8 in form of bits. The data comprises of 8 bytes each having 8 bits the position of the high bit determines the button of ps2 pressed. Out of these 8 bytes the 3rd and 4th are reserved for motor control buttons. The button of the ps2 controller pressed can hence be determined by the microcontroller atmega8 and wirelessly transmitted to another microcontroller (atmega 128) through usart.

       Libraries for ps2 and usart32 were used.

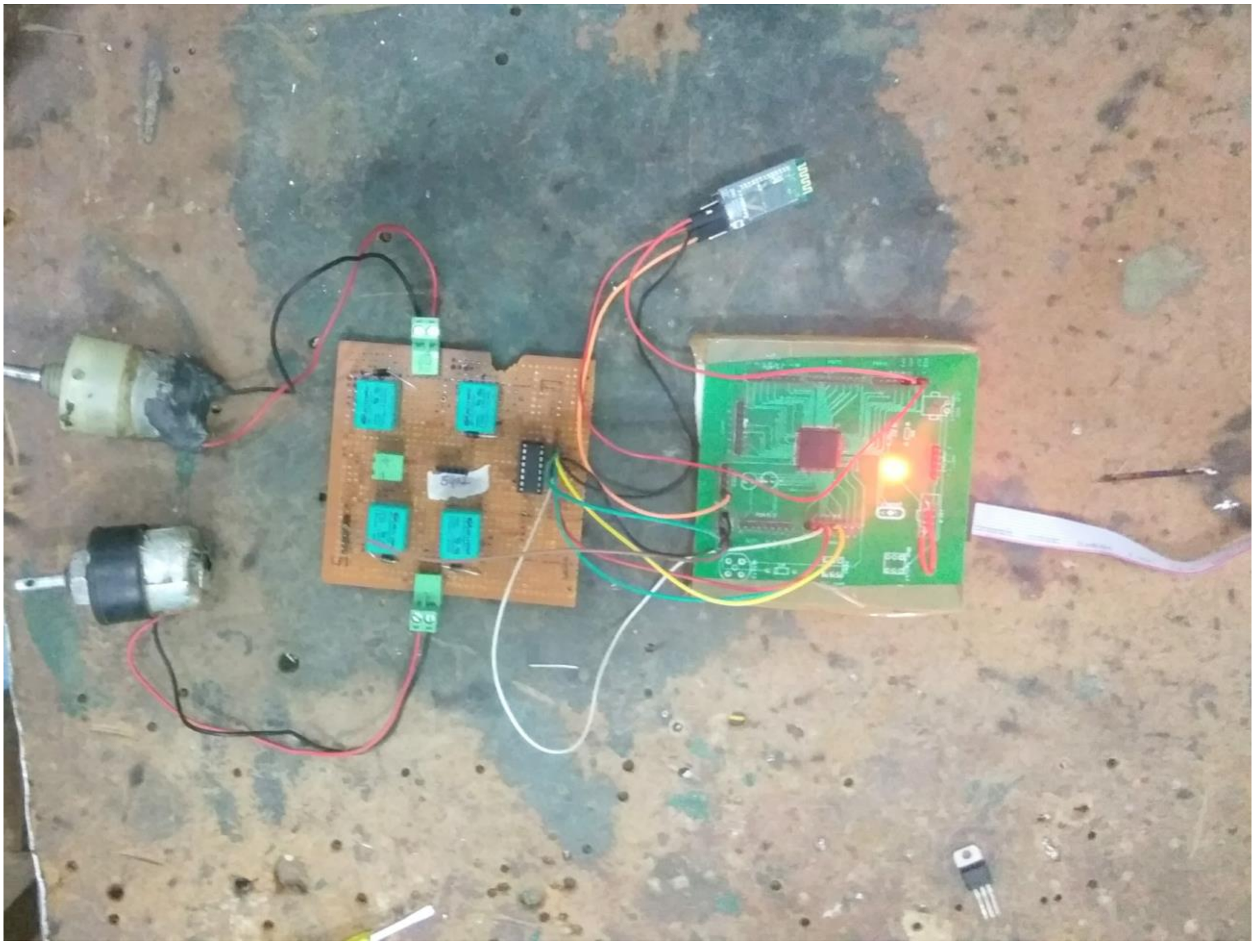


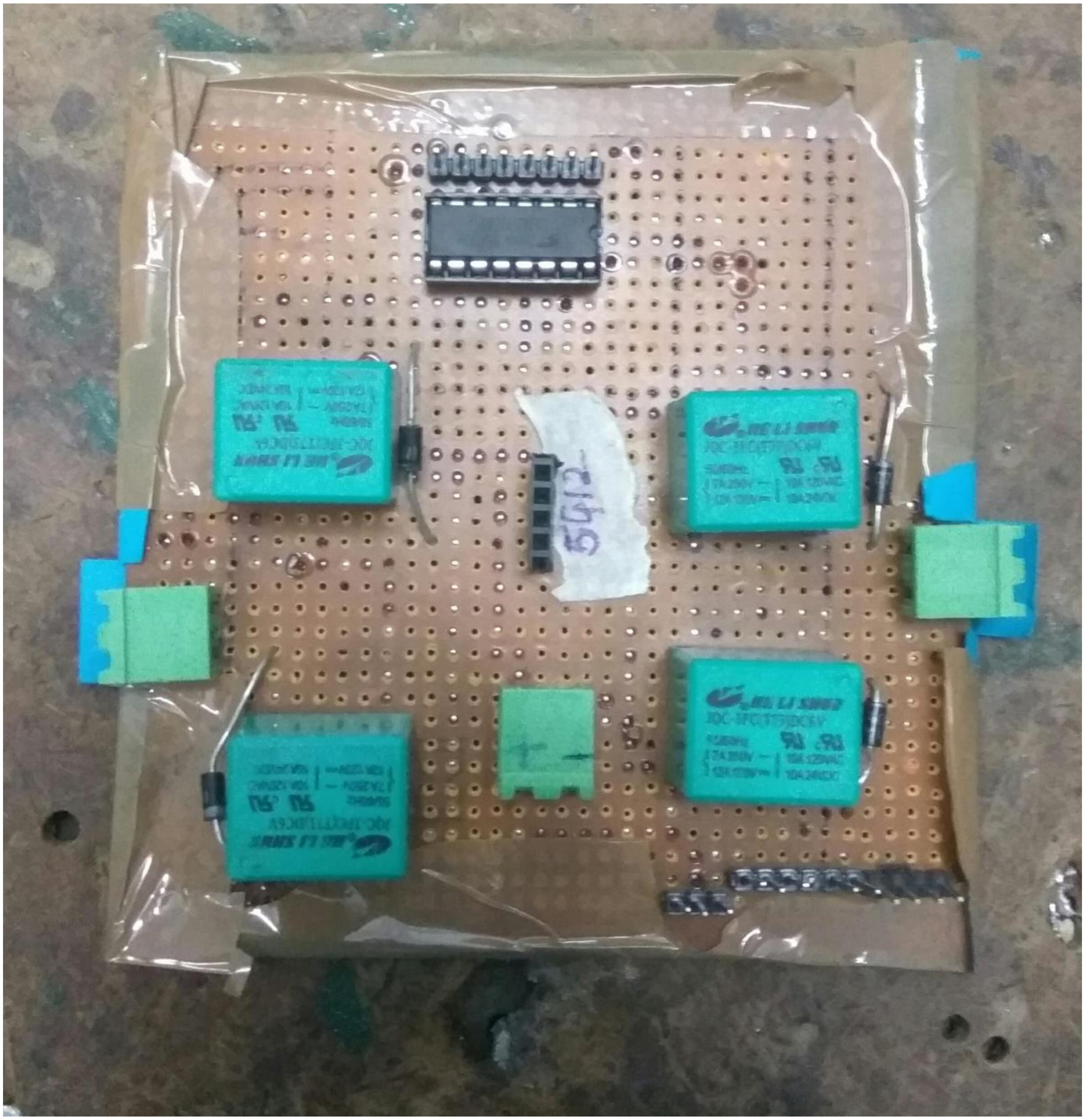


2. Bluetooth module to motors:

PS2 in total transmits 8bytes. Corresponding to buttons there are some bits in 3rd and 4th byte.Whenever a button is pressed the corresponding bit gets high, we will check this in microcontroller and if it is high then transmit some character through UART to atmega128 wirelessly through Bluetooth. Atmega128 checks the data and changes the pins high/low accordingly. Hence rotation of motor is controlled.

Output from the micro-controller can be given to the motors using motor drivers or relay switch.




5412

CODE:

1. For ps2 controller to bluetooth module through microcontroller:

```
#include <avr/io.h>


#include <avr/interrupt.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include "USART_32.h"


#include "PS2.h"

//#define BAUD 12                              //set after at commands mode
//#define BAUDRATE ((F_CPU)/(BAUD*16UL)-1)


enum {select, leftStick, rightStick, start, up, right, down, left}; //3rd byte
enum {leftFront2, rightFront2, leftFront1, rightFront1, triangle_up, circle_right, cross_down, square_left}; // 4th byte

enum left;
uint8_t x,y;

int isPressed(uint8_t dataByte, uint8_t dataBit) {
        return ((dataByte & (1 << dataBit)) ? 1 : 0);
}



int main(void)
{
        USART_Init(103);
        /* Replace with your application code */
        //USART_Transmitchar('A');
        //USART_Transmitchar(0x0D);
        _delay_us(50);
        init_PS2();                          //function to initialize spi
        _delay_us(2000);
        //initialise usart


        //USART_Transmitchar('M');
        //USART_Receive();
        while (1)
        {
            //USART_Transmitchar('B');
            //USART_Receive();
            //  USART_Transmitchar(0x0D);
            scan_PS2();
            _delay_us(500);

            x=~data_array[3];
            y=~data_array[4];
            //USART_Transmitchar('A');
            //USART_TransmitBinary(x);
            if (isPressed(x,up))     //x refers to the array which will be in terms of 000x
0000 if switch corresponding to up is pressed and up refers to enum
            {

                USART_Transmitchar('U');
                //_delay_us(50);
                //USART_Transmitchar(0x0D);
                //USART_Receive();
            }
            else if (isPressed(x,right))      //x refers to the array which will be in terms
of 000x 0000 if switch corresponding to up is pressed and up refers to enum
            {

                USART_Transmitchar('R');
                //_delay_us(50);
                //USART_Transmitchar(0x0D);
                //USART_Receive();
            }
            else if (isPressed(x,left))        //x refers to the array which will be in terms
of 000x 0000 if switch corresponding to up is pressed and up refers to enum
            {

                USART_Transmitchar('L');
                //_delay_us(50);
                //USART_Transmitchar(0x0D);
                //USART_Receive();
            }
            else if (isPressed(x,down))     //x refers to the array which will be in terms
of 000x 0000 if switch corresponding to up is pressed and up refers to enum
            {
```

```
            USART_Transmitchar('D');
            //_delay_us(50);
            //USART_Transmitchar(0x0D);
            //USART_Receive();
        }
        if (isPressed(y,triangle_up))
        {
            USART_Transmitchar('T');
            //_delay_us(50);
            //USART_Transmitchar(0x0D);
        }
        else if (isPressed(y,circle_right))
        {
            USART_Transmitchar('C');
            //_delay_us(50);
            //USART_Transmitchar(0x0D);
        }
        else if (isPressed(y,cross_down))
        {
            USART_Transmitchar('X');
            //_delay_us(50);
            //USART_Transmitchar(0x0D);
        }
        else if (isPressed(y,square_left))
        {
            USART_Transmitchar('S');
            //_delay_us(50);
            //USART_Transmitchar(0x0D);
        }
        /*else
        {
            USART_Transmitchar('e');
            _delay_us(50);
            //USART_Transmitchar(0x0D);
        }*/
    }
}
```

2. FROM BLUETOOTH MODULE TO MOTORS:

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "USART_128.h"
```

```
#define F_CPU 8000000UL

//#define baud 103
//#define BAUDRATE ((F_CPU)/(baud*16UL)-1)

#include <util/delay.h>
int main(void)
{


	DDRB |= 0xFF;


	USART_Init(103,1);
	sei();
	/* Replace with your application code */
	//USART_TransmitString("main",1);


	USART_InterruptEnable(1);
	while (1)
	{

		//USART_TransmitString("while",1);

		//USART_Transmitchar(r_data,1);



	}
}
ISR(USART1_RX_vect)
{
	unsigned char r_data;
	r_data=USART_Receive(1);
	if (r_data=='U')
	{
		USART_Transmitchar('U',1);
		//PORTB|=0b00000001;
		PORTB |= 1<< PINB0;
		//_delay_ms(50);
		//PORTB &= ~(1<< PINB0);

	}
```

```
else if(r_data=='R')
{
        USART_Transmitchar('R',1);
        //PORTB=0b00000100;
        PORTB |= 1<< PINB1;
        //_delay_ms(50);
        //PORTB &= ~(1<<PINB1);
}
else if(r_data=='L')
{
        USART_Transmitchar('L',1);
        //PORTB |=0b00000010;
        PORTB |= 1<< PINB2;
        //_delay_ms(50);
        //PORTB &= ~(1<<PINB1);
}
else if(r_data=='D')
{
        USART_Transmitchar('D',1);
        //PORTB=0b00000001;
        PORTB |= 1<< PINB3;
        //_delay_ms(50);
        //_delay_ms(10000);
        //PORTB &= ~(1<<PINB1);
}
else if(r_data=='T')
{
        USART_Transmitchar('T',1);
        //PORTB=0b00010000;
        PORTB |= 1<< PINB4;
        //_delay_ms(50);
        //PORTB &= ~(1<<PINB1);
}
else if(r_data=='C')
{
        USART_Transmitchar('C',1);
        //PORTB=0b00100000;
        PORTB |= 1<< PINB5;
        //_delay_ms(50);
        //PORTB &= ~(1<<PINB1);
}
else if(r_data=='X')
{
        USART_Transmitchar('X',1);
```

```
				//PORTB=0b01000000;
				PORTB |= 1<< PINB6;
				//_delay_ms(50);
				//PORTB &= ~(1<<PINB1);
			}
			else if(r_data=='S')
			{
				USART_Transmitchar('S',1);
				//PORTB=0b10000000;
				PORTB |= 1<< PINB7;
				//_delay_ms(50);
				//PORTB &= ~(1<<PINB1);
			}

			else
			{
				PORTB &= 0x00;
				USART_Transmitchar('N',1);
				_delay_ms(50);
			}
		}
```

BLUETOOTH INTERFACING :

The communication between the bluetooth and ATMEGA128 was done through USART(COMMUNICATION PROTOCOAL)

Before that BLUETOOTHS needs to be tested whether it is receiving and transmitting data.It can be done using TTL

Sample code has to be written for receiving and transmitting character and then BLUETOOTH has to be connected to TTL and TTL to laptop.

X-CTU software is used here to send and receive any character.

CONNECTIONS OF BLUETOOTH AND TTL :

BLUETOOTH   TTL

Rx--------Tx

Tx-----------Rx

Vcc---------Vcc

GND---------GND

(NOTE: Vcc of TTL should be connected to Vcc of bluetooth after pressing the button on the bluetooth)

***SOFTWARE**:*

X-CTU: It is application designed to enable developers to interact with RF modules.

Used to set up,configure and test the RF modules.

Install this software and drivers for TTL.

SETTING OR CHANGING THE BAUD RATE OF THE BLUETOOTHS:

Every bluetooth module has default name,password,address,mode and other properties set.

According to application these needs to be changed.

To change the properties first enter AT COMMAND mode.
Type following commands in X-CTU terminal

LIST OF COMMANDS IS MENTIONED AT THE END OF THE DOCUMENT

PROBLEMS FACED AND THEIR SOLUTIONS

1.)Problem in connecting the jumper wires with the PS2
The jumper holes were small as compared to pins in PS2 controller. So firstly we got the idea of converting data of SPI to UART through usb to serial converter but for that we had to check that what data was coming in uart for the corresponding SPI data. But that didn't work. So we decided to stick with the jumpers connected them anyhow can we checked the connection with other with other with other PS2 adaptor (male headers)and connected to laptop with USB. Windows has an inbuilt feature of checking the PS2 controller by running a command Joy.cpl.

2.) It was the time to check the the code. We interface the PS2 controller with microcontroller and connected the uart pins of MCU with TTL and connected to the laptop the laptop to the laptop USB port and checked the data in X-CTU software. But it didn't work. We also tried different Baud rates rates in the X-CTU software .So we found out that we mistook the PS2 controller to be master but it was slave and MCU was master. So finally we tried with 8 MHz fcpu and 4800 baud rate in code for UART initialization and data was coming in X-CTU.

3.) Problem in pairing two bluetooth modules.
To overcome this we changed them into slave and master using AT-Command mode. Now only the specified slave can connect to the master reducing its connecting time and network . Data flow can be either way (slave ⇔ master).

4.) There was problem in connection between the two bluetooth modules.To overcome them one of the bluetooth was set as slave and other as master.Slave can get connected only to its master while master can connected to any other devices.Once the master slave is done using at commands the bluetooths get connected instantly without any disturbance.

5.) Without using INTERRUPTS in the code processing stopped in between so it is adviSable to use interrupts.

6.)BAUD RATES of both the bluetooths and that written in the code should be same to avoid no output.

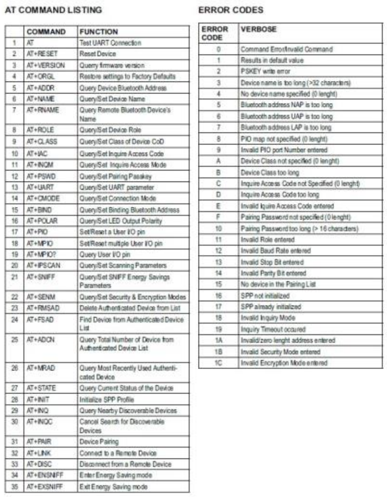
References

SPI (serial peripheral interfacing)-Spi Avr Maxembedded

PS2 controller-Ps2

Softwares used

Atmel studio
X-CTU
eXtreme Burner
DipTrace



AT COMMAND LISTING

	COMMAND	FUNCTION
1	AT	Test UART Connection
2	AT+RESET	Reset Device
3	AT+VERSION	Query firmware version
4	AT+ORGL	Restore settings to Factory Defaults
5	AT+ADDR	Query Device Bluetooth Address
6	AT+NAME	Query/Set Device Name
7	AT+RNAME	Query Remote Bluetooth Device's Name
8	AT+ROLE	Query/Set Device Role
9	AT+CLASS	Query/Set Class of Device CoD
10	AT+IAC	Query/Set Inquire Access Code
11	AT+INQM	Query/Set Inquire Access Mode
12	AT+PSWD	Query/Set Pairing Passkey
13	AT+UART	Query/Set UART parameter
14	AT+CMODE	Query/Set Connection Mode
15	AT+BIND	Query/Set Binding Bluetooth Address
16	AT+POLAR	Query/Set LED Output Polarity
17	AT+PIO	Set/Reset a User I/O pin
18	AT+MPIO	Set/Reset multiple User I/O pin
19	AT+MPIO?	Query User I/O pin
20	AT+IPSCAN	Query/Set Scanning Parameters
21	AT+SNIFF	Query/Set SNIFF Energy Savings Parameters
22	AT+SENM	Query/Set Security & Encryption Modes
23	AT+RMSAD	Delete Authenticated Device from List
24	AT+FSAD	Find Device from Authenticated Device List
25	AT+ADCN	Query Total Number of Device from Authenticated Device List
26	AT+MRAD	Query Most Recently Used Authenti-cated Device
27	AT+STATE	Query Current Status of the Device
28	AT+INIT	Initialize SPP Profile
29	AT+INQ	Query Nearby Discoverable Devices
30	AT+INQC	Cancel Search for Discoverable Devices
31	AT+PAIR	Device Pairing
32	AT+LINK	Connect to a Remote Device
33	AT+DISC	Disconnect from a Remote Device
34	AT+ENSNIFF	Enter Energy Saving mode
35	AT+EXSNIFF	Exit Energy Saving mode

ERROR CODES

ERROR CODE	VERBOSE
0	Command Error/Invalid Command
1	Results in default value
2	PSKEY write error
3	Device name is too long (>32 characters)
4	No device name specified (0 lenght)
5	Bluetooth address NAP is too long
6	Bluetooth address UAP is too long
7	Bluetooth address LAP is too long
8	PIO map not specified (0 lenght)
9	Invalid PIO port Number entered
A	Device Class not specified (0 lenght)
B	Device Class too long
C	Inquire Access Code not Specified (0 lenght)
D	Inquire Access Code too long
E	Invalid Iquire Access Code entered
F	Pairing Password not specified (0 lenght)
10	Pairing Password too long (> 16 characters)
11	Invalid Role entered
12	Invalid Baud Rate entered
13	Invalid Stop Bit entered
14	Invalid Parity Bit entered
15	No device in the Pairing List
16	SPP not initialized
17	SPP already initialized
18	Invalid Inquiry Mode
19	Inquiry Timeout occured
1A	Invalid/zero lenght address entered
1B	Invalid Security Mode entered
1C	Invalid Encryption Mode entered