***Wireless driving shield***

***Team members***:. ***Mentors***

Saurabh Pranjale. Sushen Kashyap

Dhruv Patel. Divya Poddar

Mansi Sampat

Milie Bhanani***Index***

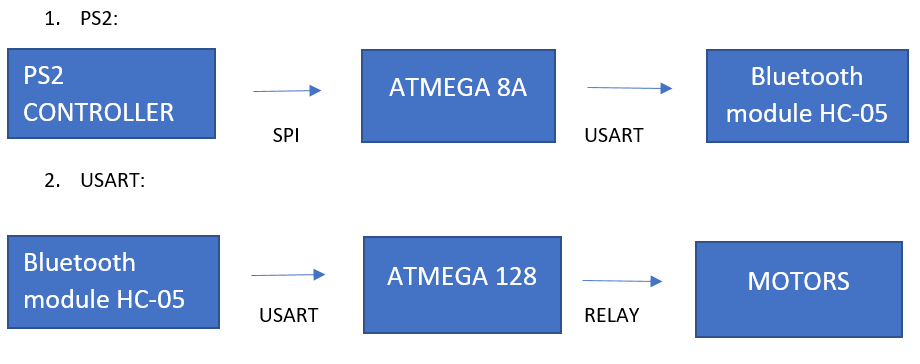
1. ***Project abstract***
2. ***Motivation***
3. ***Working***
4. ***Bluetooth interfacing***
5. ***Problems faced and their solutions***
6. ***References***
7. ***Softwares used***

‌***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.

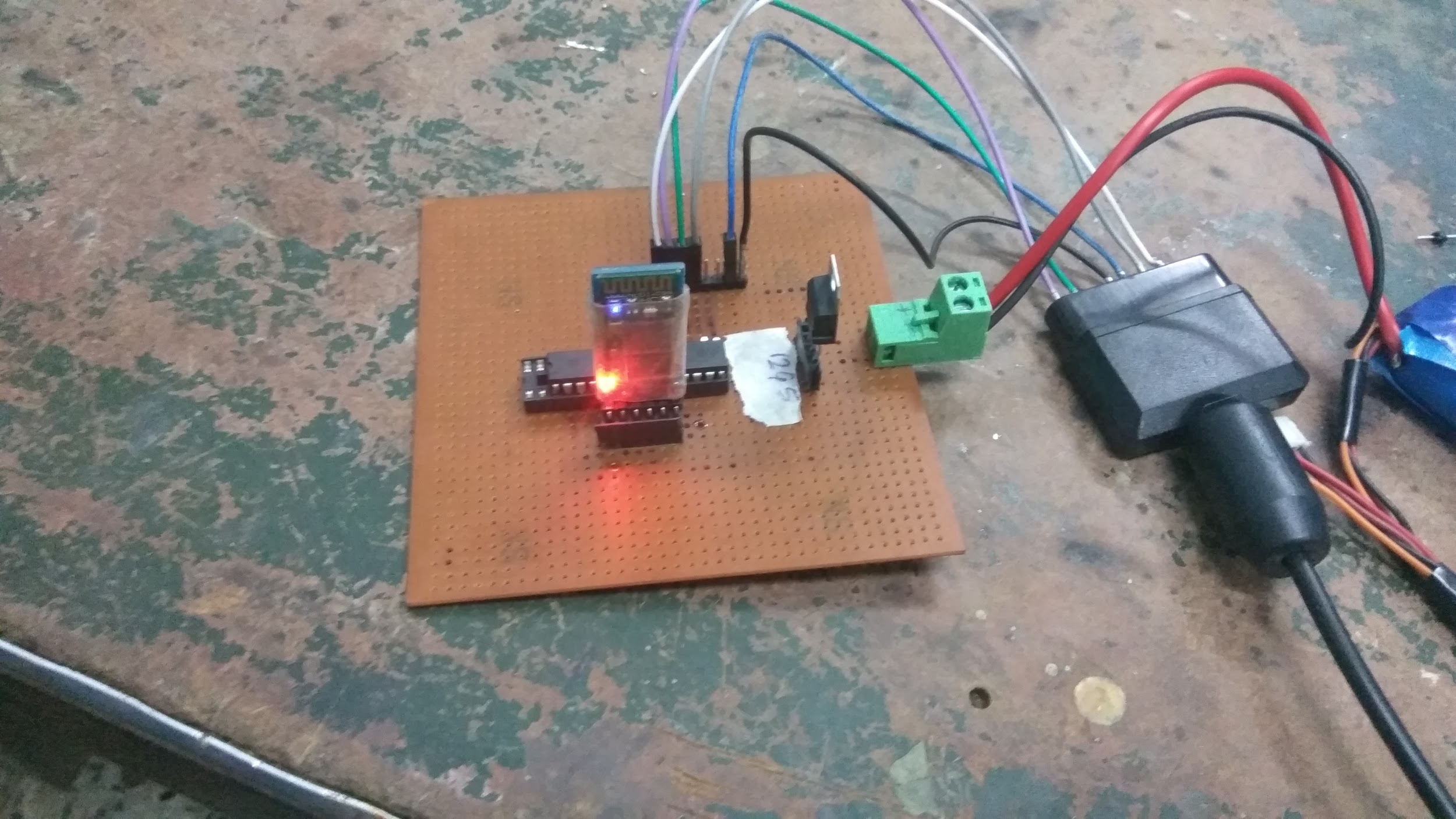
The project was divided into two modules:

  
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***

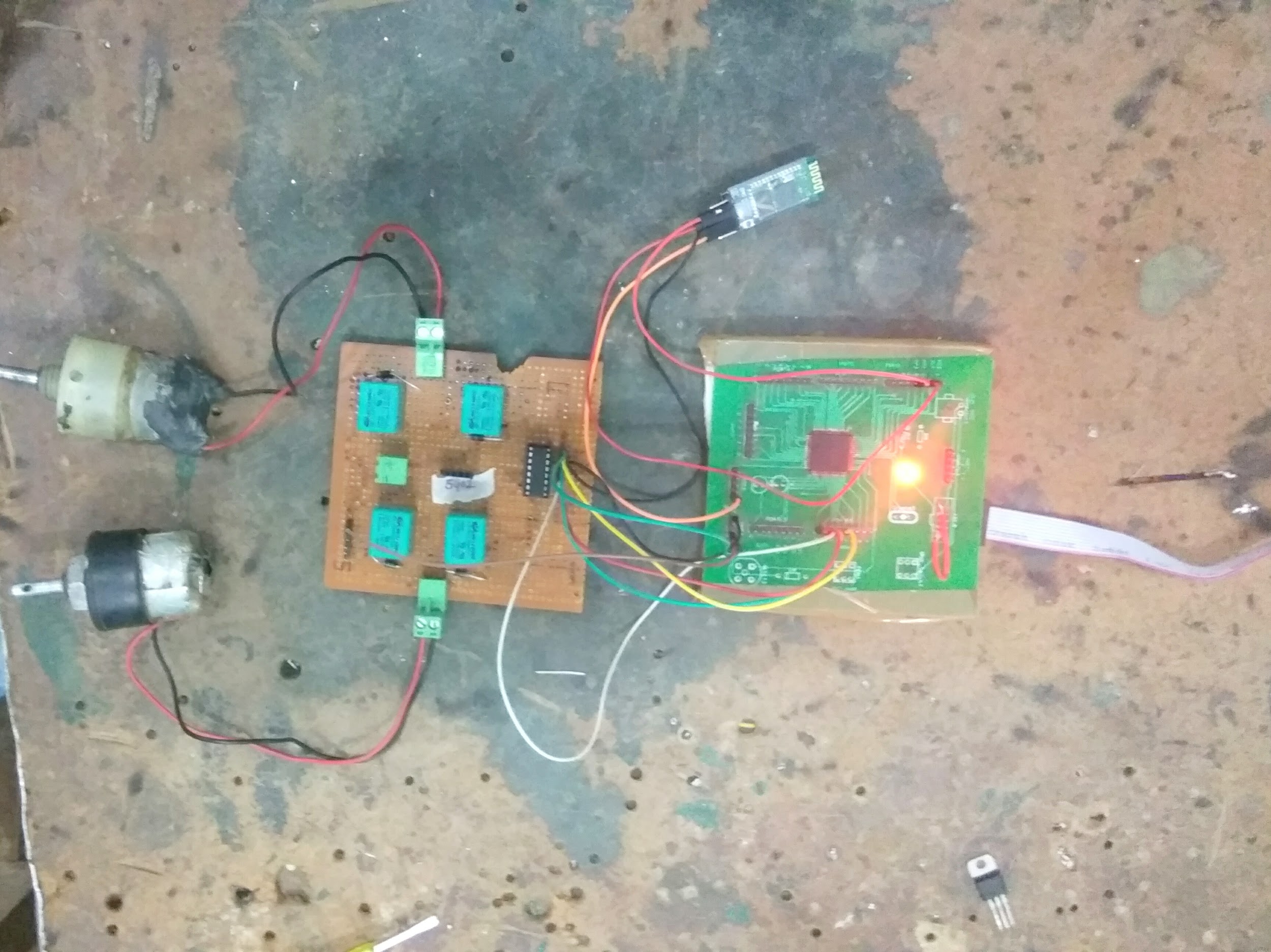
1. PS2 to atmega8:
   1. Data is sent from ps2 to atmega 8 in form of bits. The data comprises of 8 bytes each having 8 bits the position of the high bit determines the button of ps2 pressed. Out of these 8 bytes the 3rd and 4th are reserved for motor control buttons. The button of the ps2 controller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

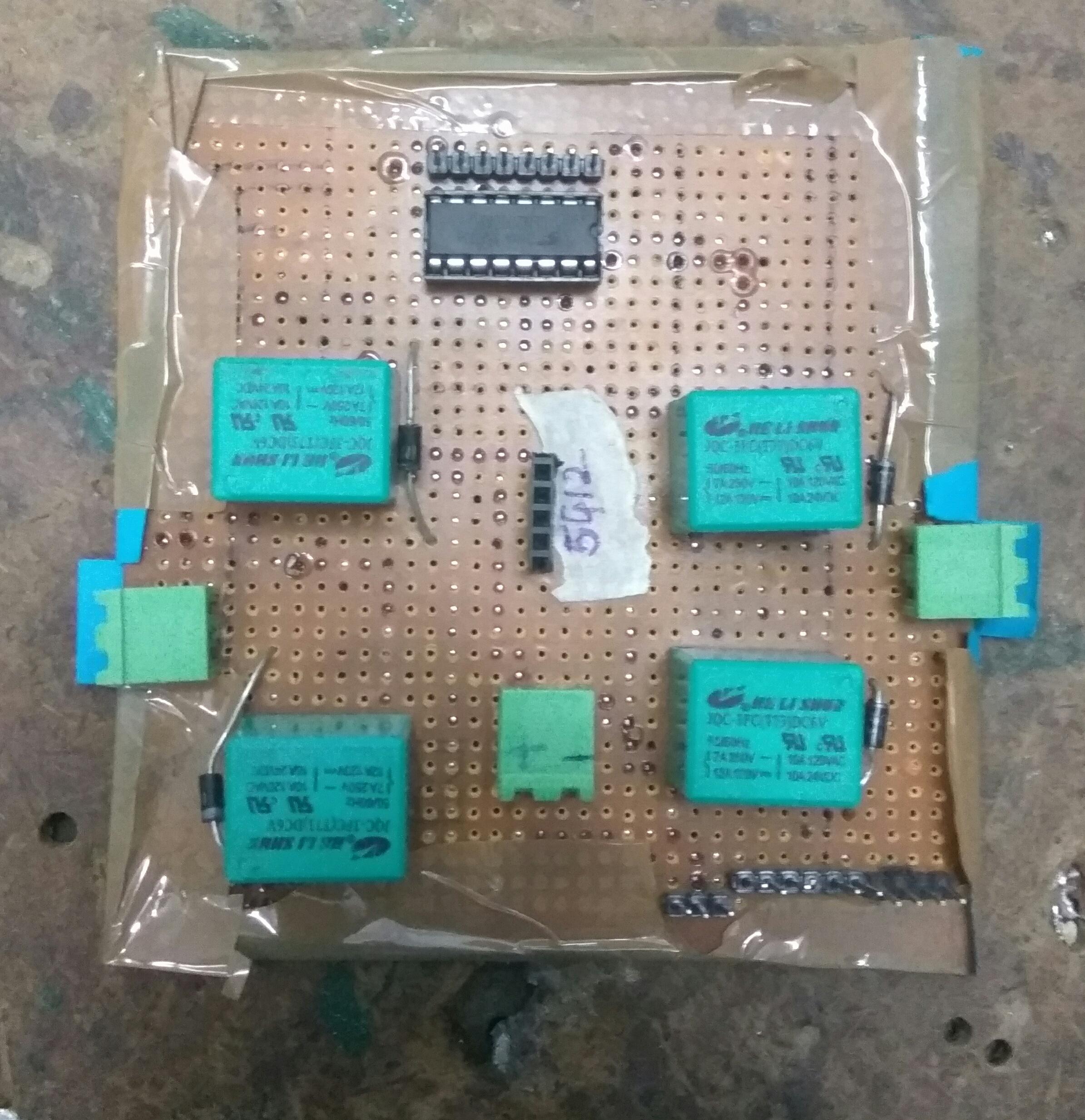
Libraries for ps2 and usart32 were used. 



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.  




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](http://maxembedded.com/2013/11/the-spi-of-the-avr/)  
Maxembedded

PS2 controller-[Ps2](http://store.curiousinventor.com/guides/PS2/)

‌***Softwares used***

Atmel studio

X-CTU

eXtreme Burner

DipTrace  
  
