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

#ifndef F_CPU
//define cpu clock speed if not defined
#define F_CPU 16000000UL
#endif

//set desired baud rate
#define BAUDRATE 1200

//calculate UBRR value
#define UBRRVAL ((F_CPU/(BAUDRATE*16UL))-1)

//define receive parameters
#define SYNC 0XAA// synchro signal
#define RADDR 0xBB

#define TURN_LEFT 0x00
#define STOP_TURN_LEFT 0x01

#define TURN_RIGHT 0x02
#define STOP_TURN_RIGHT 0x03

#define MOVE_FORWARD 0x04
#define STOP_MOVE_FORWARD 0x05

#define MOVE_BACKWARD 0x06
#define STOP_MOVE_BACKWARD 0x07

#define MAX_TRANSMISSIONS 100

void USART_Init(void)
{
//Set baud rate
UBRRL=(uint8_t)UBRRVAL; //low byte
UBRRH=(UBRRVAL>>8); //high byte

//Set data frame format: asynchronous mode,no parity, 1 stop bit, 8 bit size
UCSRC= (1<<URSEL)| (0<<UMSEL) | (0<<UPM1)  | (0<<UPM0)|
      (0<<USBS) | (0<<UCSZ2) | (1<<UCSZ1) | (1<<UCSZ0);

//Enable Transmitter and Receiver and Interrupt on receive complete
UCSRB=(1<<TXEN);
}
```

```
void USART_vSendByte(uint8_t u8Data)
{
// Wait if a byte is being transmitted
   while((UCSRA&(1<<UDRE)) == 0);

// Transmit data
UDR = u8Data;
}

void Send_Packet(uint8_t addr, uint8_t cmd)
{
USART_vSendByte(SYNC);//send synchro byte
USART_vSendByte(addr);//send receiver address
USART_vSendByte(cmd);//send increment command
USART_vSendByte((addr+cmd));//send checksum
}

int main(void)
{
int i;
// Initialize USART
USART_Init();

DDRB = 0x00; // B0 as input
PORTB = 0xFF; // pull - up on port B

// Setup the led from the uC - PD7 pin
DDRD |= (1 << PD7);
PORTD &= ~(1 << PD7); // Initially OFF

while(1)
{
if ((PINB & 0x01) == 0x00) {
PORTD |= (1 << PD7);

// Send the Turn Left Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, TURN_LEFT);
// Wait until the buttor is released
while ((PINB & 0x01) == 0x00);

// Send the Stop Turn Left Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, STOP_TURN_LEFT);
```

```
PORTD &= ~(1 << PD7);
}

if ((PINB & 0x04) == 0x00) {
PORTD |= (1 << PD7);

// Send the Turn Right Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, TURN_RIGHT);
// Wait until the buttor is released
while ((PINB & 0x04) == 0x00);

// Send the Stop Turn Right Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, STOP_TURN_RIGHT);

PORTD &= ~(1 << PD7);
}

if ((PINB & 0x02) == 0x00) {
PORTD |= (1 << PD7);

// Send the Move Forward Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, MOVE_FORWARD);
// Wait until the buttor is released
while ((PINB & 0x02) == 0x00);

// Send the Stop Move Forward Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, STOP_MOVE_FORWARD);

PORTD &= ~(1 << PD7);
}

if ((PINB & 0x08) == 0x00) {
PORTD |= (1 << PD7);

// Send the Move Backwords Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, MOVE_BACKWARD);
// Wait until the buttor is released
while ((PINB & 0x08) == 0x00);
```

```
// Send the Stop Move Backwords Command
for (i = 0; i < MAX_TRANSMISSIONS; i++)
Send_Packet(RADDR, STOP_MOVE_BACKWARD);

PORTD &= ~(1 << PD7);
}
}
return 0;
}
```