Smart Home Automation Project

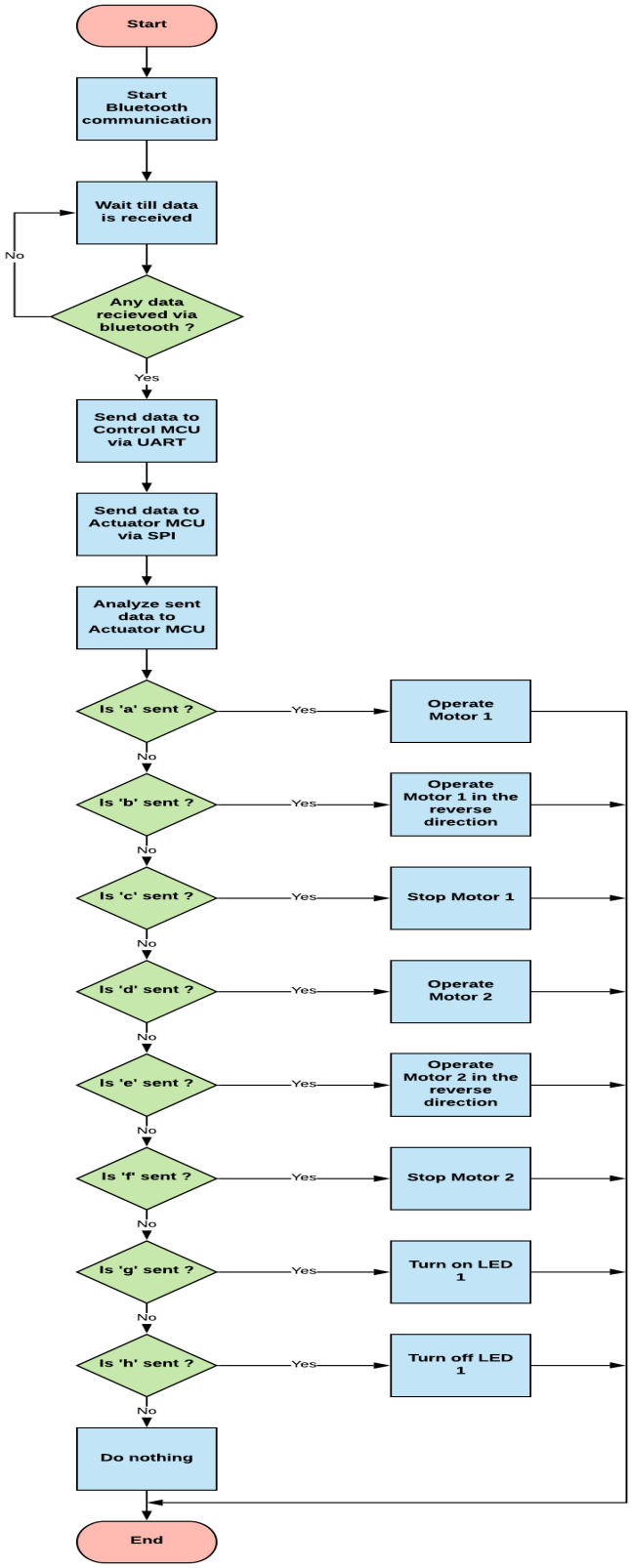
Submitted by:

Name: Nouran Adel Hassan Ahmed

Group: M12-online

Table of contents

Flow Chart:


Start
Start Bluetooth communication
Wait till data is received
No
Any data recieved via bluetooth ?
Yes
Send data to Control MCU via UART
Send data to Actuator MCU via SPI
Analyze sent data to Actuator MCU
Is 'a' sent ?
Yes
Operate Motor 1
No
Is 'b' sent ?
Yes
Operate Motor 1 in the reverse direction
No
Is 'c' sent ?
Yes
Stop Motor 1
No
Is 'd' sent ?
Yes
Operate Motor 2
No
Is 'e' sent ?
Yes
Operate Motor 2 in the reverse direction
No
Is 'f' sent ?
Yes
Stop Motor 2
No
Is 'g' sent ?
Yes
Turn on LED 1
No
Is 'h' sent ?
Yes
Turn off LED 1
No
Do nothing
End

Description:

This is a home automation project in which one can control whatever devices or functionalities they want using their mobile phones.

The project uses the following modules:

- Bluetooth module
- Mobile phone for communication
- 2 Microcontrollers
- H Bridge
- 2 DC Motors
- Resistor
- LED

The project works as follows:

At the beginning the UART is initialized with a certain baud rate so that the communication between the Bluetooth module and the Control Microcontroller is established. Then the user uses a mobile application to send the required data to the Bluetooth module. This data is then sent to the Control Microcontroller via UART. Control Microcontroller initializes SPI and sends this data to Actuator Microcontroller via SPI. Actuator Microcontroller then analyzes the received data and performs the required functionalities accordingly where:

- If the received is character a, Motor 1 operates.
- If the received is character b, Motor 1 operates in the reverse direction.
- If the received is character c, Motor 1 stops.
- If the received is character d, Motor 2 operates.
- If the received is character e, Motor 2 operates in the reverse direction.
- If the received is character f, Motor 2 stops.
- If the received is character g, LED 1 is turned on.
- If the received is character h, LED 1 is turned off.
- If the received is anything else, nothing is done.

The executed functionality keeps working till the project stops running.

Code Architecture:

The project is divided into two main projects, one for the Control MCU and the other for Actuator MCU.

The project follows the layered architecture which has 3 layers:

- APP
- HAL
- MCAL

The APP layer contains the code of the application, while the HAL has the codes of the hardware modules used, and the MCAL has the codes of the MCU modules used.

Each of the 2 projects of Control MCU and Actuator MCU has 3 layers, each contains the used modules.

Each module whether in APP, HAL or MCAL has 3 files: interface file which contains the used macros along with functions' prototypes, private file which contains the private registers used, and program file which contains the functions' definitions.

Application of Control MCU:

```c
#include "LBIT_MATH.h"
#include "LSTD_TYPES.h"
#include "MUART_interface.h"
#include "MUART_private.h"
#include "MDIO_interface.h"
#include "MSPI_interface.h"
#include "HBLUETOOTH_interface.h"

#define F_CPU 16000000UL
#include "util/delay.h"


int main(void)
{

    u8_t recv_var = 0;

    muart_init(MUART_9600_BPS);

    mspi_init(MSPI_SAMPLE_R_SETUP_F, MSPI_PRE_32);

    /* Replace with your application code */
    while (1)
    {

        muart_recvByte(&recv_var);

        mspi_sendRecvByte(recv_var);

        _delay_ms(1000);

    }
}
```

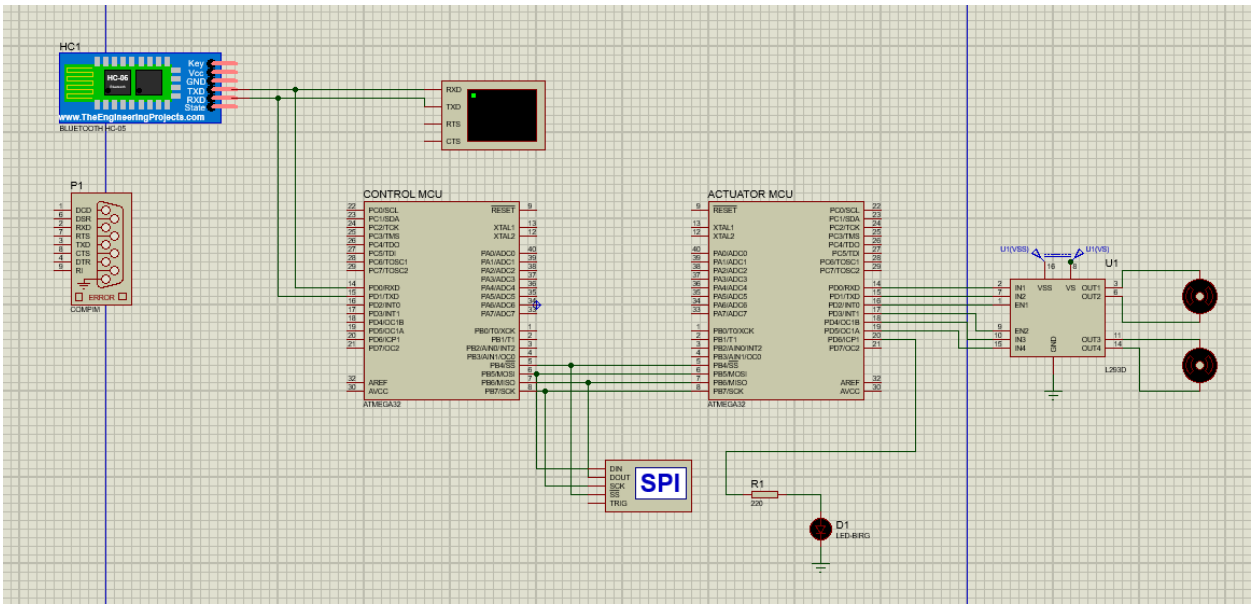
Application of Actuator MCU:

```c
#include "LBIT_MATH.h"
#include "LSTD_TYPES.h"
#include "MDIO_interface.h"
#include "MSPI_interface.h"
#include "HDC_MOTOR_interface.h"
#include "HLED_interface.h"


#define F_CPU 16000000UL
#include "util/delay.h"

int main(void)
{

    mdio_pinStatus(PORTB, PIN4, INPUT_FLOAT);
    mdio_pinStatus(PORTB, PIN5, INPUT_FLOAT);
    mdio_pinStatus(PORTB, PIN6, OUTPUT);
    mdio_pinStatus(PORTB, PIN7, INPUT_FLOAT);

    mspi_init_slave(MSPI_SAMPLE_R_SETUP_F, MSPI_PRE_32);

    hdc_motor1_init();
    hdc_motor2_init();
    hled_init(LED1);
```

```
/* Replace with your application code */
while (1)
{

    if (mspi_RecvByte_slave() == (char)'a')
    {
        hdc_motor1_operate_dir1();
    }
    else if (mspi_RecvByte_slave() == (char)'b')
    {
        hdc_motor1_operate_dir2();
    }
    else if(mspi_RecvByte_slave() == (char)'c')
    {
        hdc_motor1_stop();
    }
    else if(mspi_RecvByte_slave() == (char)'d')
    {
        hdc_motor2_operate_dir1();
    }
    else if(mspi_RecvByte_slave() == (char)'e')
    {
        hdc_motor2_operate_dir2();
    }
    else if(mspi_RecvByte_slave() == (char)'f')
    {
        hdc_motor2_stop();
    }
    else if(mspi_RecvByte_slave() == (char)'g')
    {
        hled_ledON(LED1);
    }
    else if(mspi_RecvByte_slave() == (char)'h')
    {
        hled_ledOFF(LED1);
    }
    else
    {
        /*Do nothing*/
    }

    _delay_ms(1000);

}
```

Simulation:

Simulation is done using Proteus 8. Following is the schematic:



Before starting the project the Bluetooth of the mobile phone of the user should be paired with the Bluetooth of the laptop and the working COM port is modified in the Bluetooth module used in Proteus .

Any mobile app for sending data to Bluetooth module can be used like: Blueterm, Arduino Bluetooth Home Automation, …etc.

Virtual Terminal and SPI Debugger are used to test the data sent from Bluetooth module to Control MCU and from Control MCU to Actuator MCU.

COMPIM is also added in case the Bluetooth module fails to work then this virtual Bluetooth can connected and used in this case.

Connections:

- Bluetooth RXD is connected to Control MCU TXD (PD1) and Virtual Terminal TXD.
- Bluetooth TXD is connected to Control MCU RXD (PD0) and Virtual Terminal RXD.
- Control MCU SS is connected to Actuator MCU SS (PB4) and SPI Debugger SS.

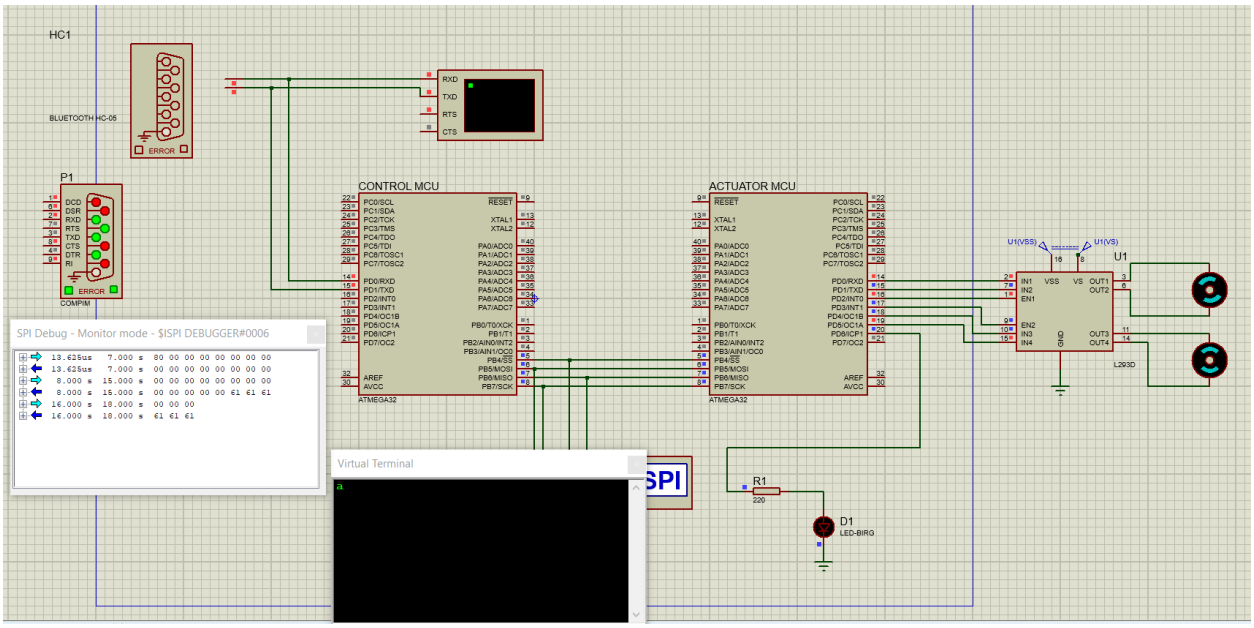
- Control MCU MOSI is connected to Actuator MCU MOSI (PB5) and SPI Debugger DIN.
- Control MCU MISO is connected to Actuator MCU MISO (PB6) and SPI Debugger DOUT.
- Control MCU SCK is connected to Actuator MCU SCK (PB7) and SPI Debugger SCK.
- Actuator MCU PD0 is connected to H-Bridge IN1.
- Actuator MCU PD1 is connected to H-Bridge IN2.
- Actuator MCU PD2 is connected to H-Bridge EN1.
- Actuator MCU PD3 is connected to H-Bridge EN2.
- Actuator MCU PD4 is connected to H-Bridge IN3.
- Actuator MCU PD5 is connected to H-Bridge IN4.
- H-Bridge VSS is connected to 5V.
- H-Bridge VS is connected to 12V.
- H-Bridge GND is connected to Ground.
- H-Bridge OUT1 is connected to a terminal of Motor 1.
- H-Bridge OUT2 is connected to the other terminal of Motor 1.
- H-Bridge OUT3 is connected to a terminal of Motor 2.
- H-Bridge OUT4 is connected to the other terminal of Motor 2.
- Actuator MCU PD6 is connected to a resistor which is connected to a LED that is connected to the Ground.

Modules used in simulation:

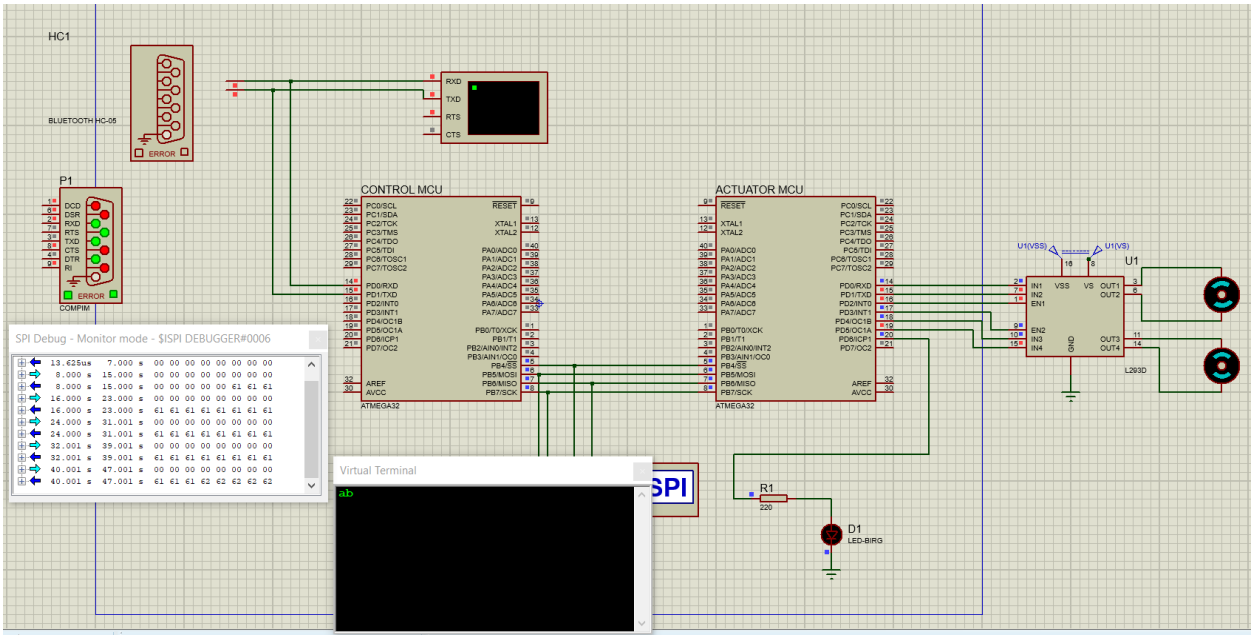
- Bluetooth is HC-05.
- MCU is Atmega 32a.
- Resistor is 220 ohm.
- H-Bridge is L293D.
- Motor is a simple DC Motor.

Results:

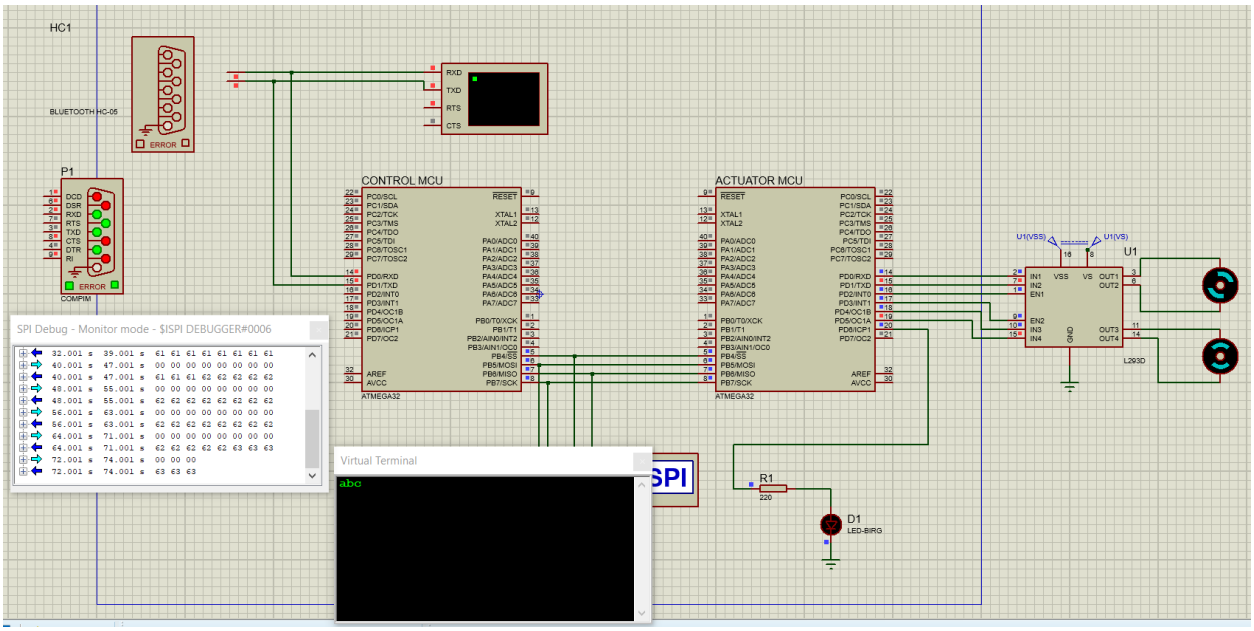
When sending ‘a’, Motor 1 operates:



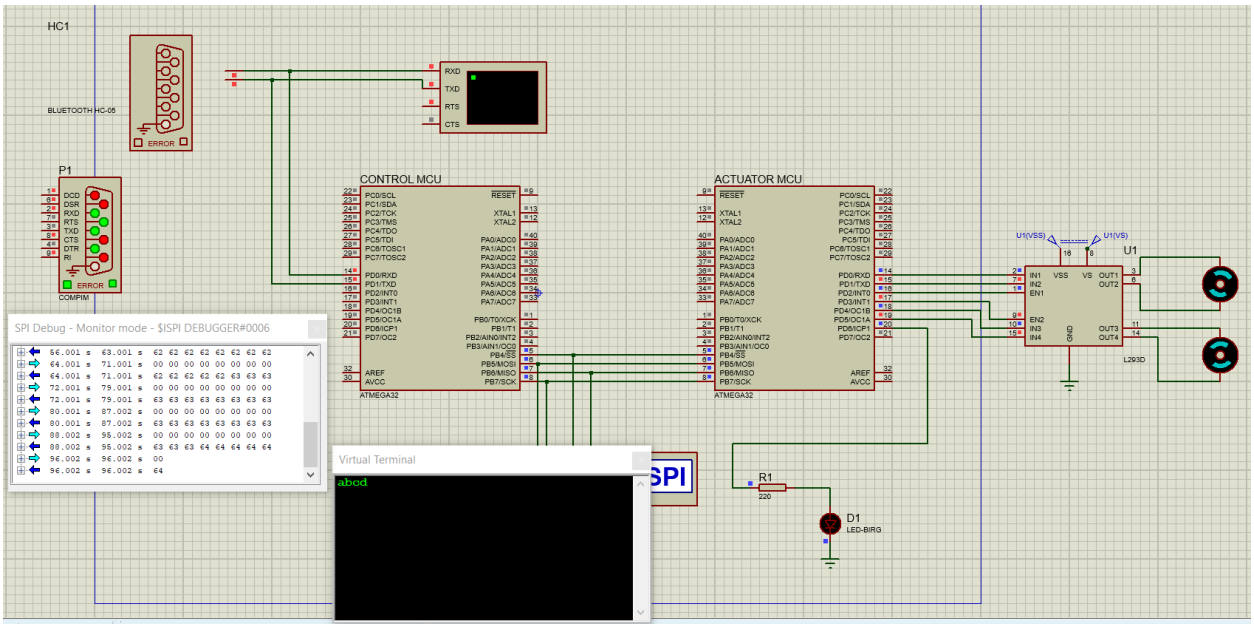
When sending ‘b’, Motor 1 operates in the reverse direction:



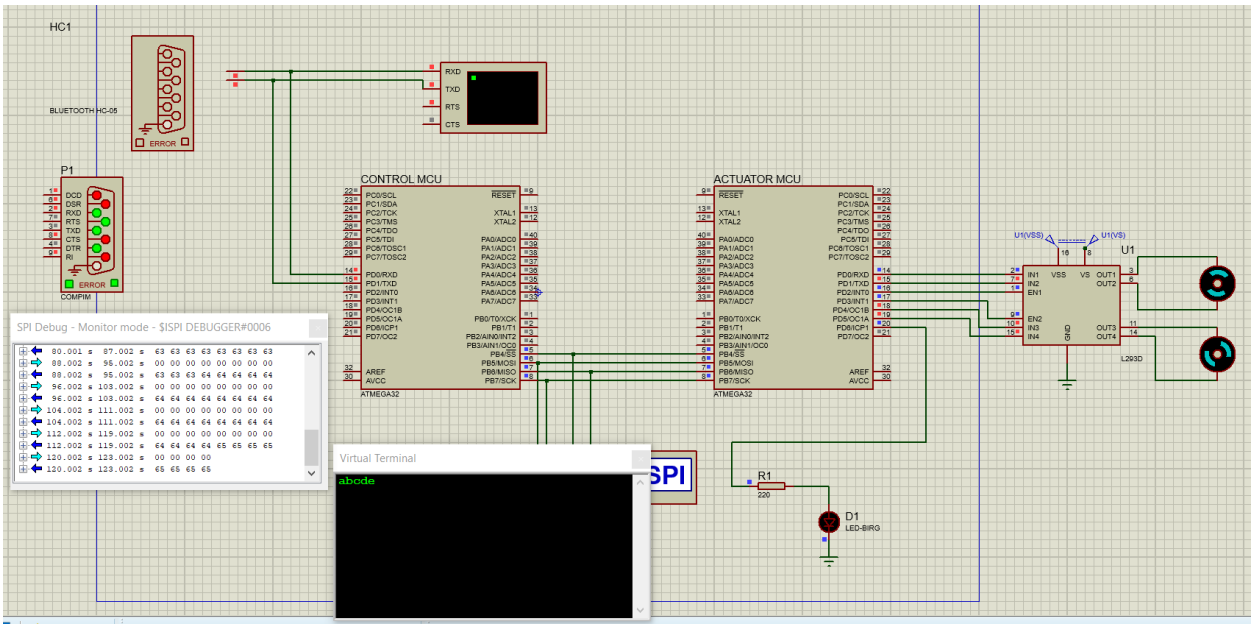
When sending ‘c’, Motor 1 stops:



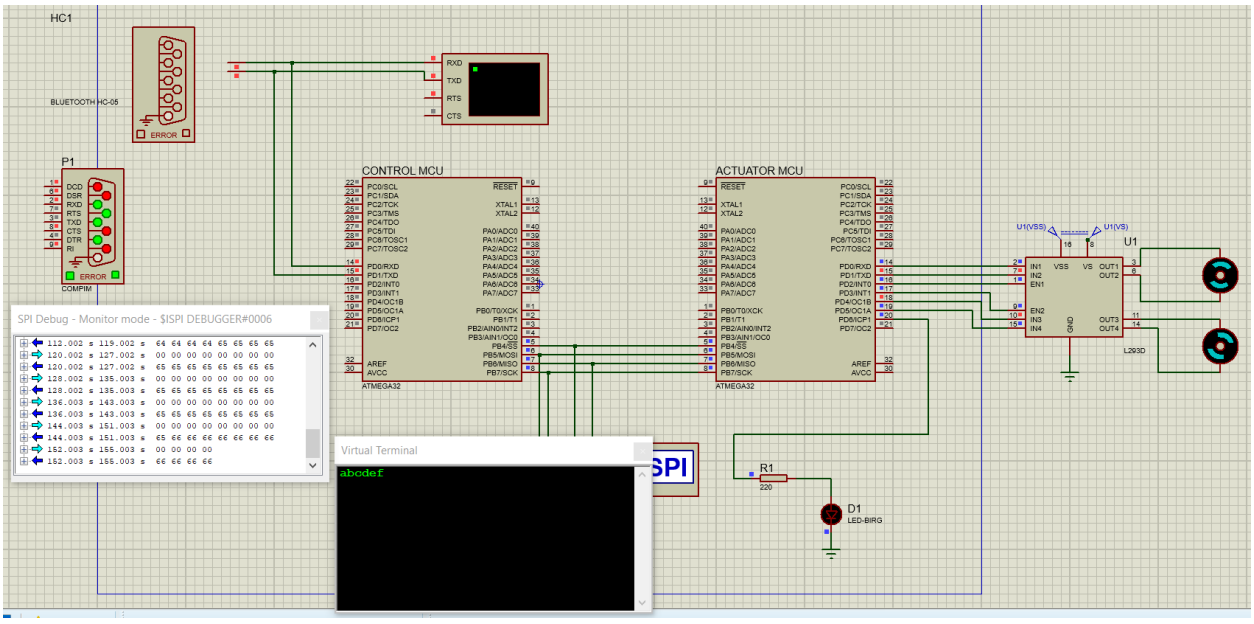
When sending ‘d’, Motor 2 operates:



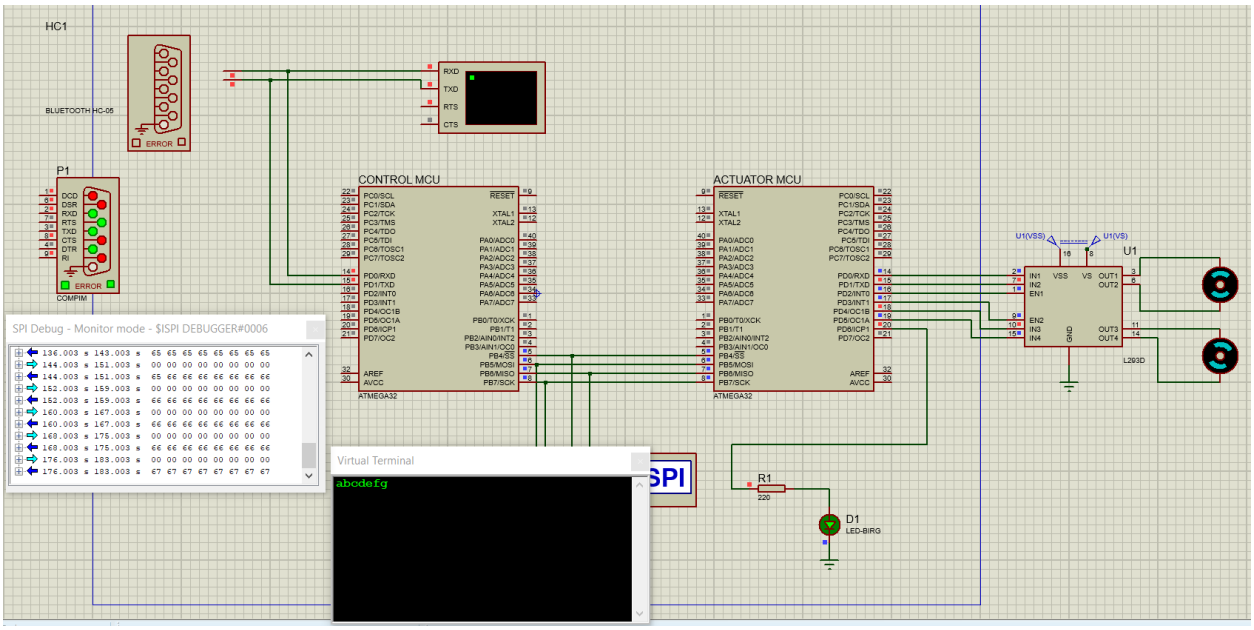
When sending ‘e’, Motor 2 operates in the reverse direction:



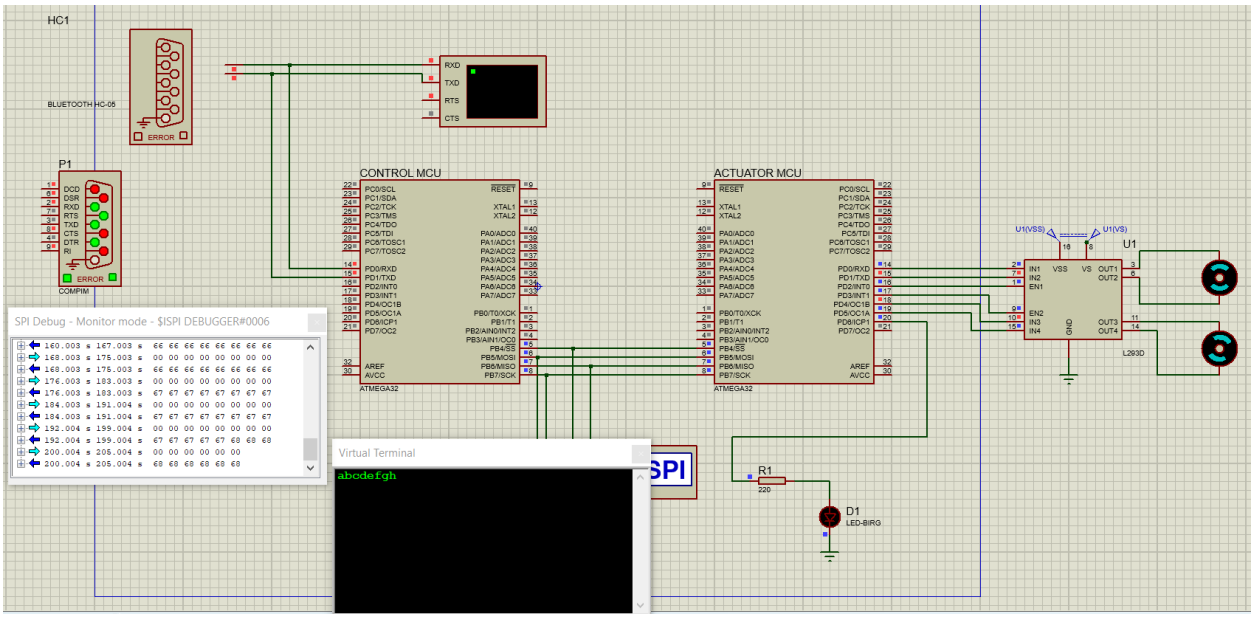
When sending ‘f’, Motor 2 stops:



When sending ‘g’, LED 1 is turned on:



When sending ‘h’, LED 1 is turned off:



Conclusion:

The project works as desired and the results are correct. It can be used to operate any other desired modules.