Hacettepe UNIVERSITY

Electrical and Electronics Engineering

417 Term Project

May 27, 2022

Mehmet Alperen Şahin, 21806955

Mustafa Yasin Ateş, 21889852

Contents

Abstract

The aim of the project was building a solid, efficient, smart attendance system with temperature control to avoid new Covid-19 cases and create a solid medium between professor and students.

We started to project with another member of our group but unfortunately he quit right before the submission data thus, we needed to remove the RFID part of the system to speed up the process.

1 Introduction

1.1 Equipments

We started with planing the system and choosing equipments. We choose DS18B20 as sensor, ESP8266 to connect wifi, RC522 RFID module to get user id and 16x2 LCD screen to preview data. Further information about why and how will be included in next section.

1.2 Temperature Sensor

Alperen used LM35, DHT-11 but they weren't stable enough to measure human temperature therefore Alperen moved with DS18B20 to get stable data in small time interval.

1.3 ESP8266

Alperen used AT commands to connect wifi, broker, publish data and subscribe that data. We could have used NodeMCU or we could have at least flashed a program into ESP8266 to avoid using AT commands.

1.4 RFID

Secondly Alperen used RFID module to get user id hopefully our school id cards were also 13.56MHz therefore we were able to get school id from it.

1.5 LCD

Thirdly Yasin soldered the LCD with headers and made the connections. Yasin used a 10k potentiometer to control backlit of the LCD screen.

Whole system was scattered but working.

2 Challenges

The whole project was concluded by 2 members of 3 at day before due date. RFID, LCD, temperature sensor was working just ok but mqtt servers were usually slow to respond and callback data wasn't really good enough to use.

3 Code

3.1 Temperature Sensor

Alperen used ADC registers to activate analog to digital converters and read data back from the sensor.

```
ADC10CTL0 = SREF_0 + ADC10SHT_3 + ADC10ON;
ADC10CTL1 = INCH_4 | SHS_0 | ADC10DIV_0 | CONSEQ_0;
ADC10AE0 |= BIT4;
```

3.2 ESP8266

To connect the wifi we pass ssid and password of the network to ConnectWifi function. This function transmits AT commands to the ESP8266 by UART.

AT+RST command is send to reset to esp8266 every time we restart to system. AT command is send to check whether esp8266 is alive or not. AT+CWMODE sets the esp8266 station or ap depending on the integer we pass. AT+CWJAP is responsible for connecting the wifi.

```
void ConnectWifi(char *ssid ,char *password){

  UARTTransmitString("AT+RST\r\n",(unsigned short)strlen("AT+RST\r\n"));
  Enable_RX_Interrupt;
  MSDelay(15);
  while(!(IsReceiveSendOK())){
      UARTTransmitString("AT+RST\r\n",(unsigned short)strlen("AT+RST\r\n"));
      Enable_RX_Interrupt;
      MSDelay(15);
  }
  MSDelay(1000);

  UARTTransmitString("AT\r\n",(unsigned short)strlen("AT\r\n"));
  Enable_RX_Interrupt;
  MSDelay(15);
  while(!(IsReceiveSendOK())){
      UARTTransmitString("AT\r\n",(unsigned short)strlen("AT\r\n"));
      Enable_RX_Interrupt;
      MSDelay(15);
  }

  UARTTransmitString("AT+CWMODE=1\r\n",(unsigned short)strlen("AT+CWMODE=1\r\n"));
  Enable_RX_Interrupt;
  MSDelay(10);
  while(!(IsReceiveSendOK())){
      UARTTransmitString("AT+CWMODE=1\r\n",(unsigned short)strlen("AT+CWMODE=1\r\n"));
      Enable_RX_Interrupt;
      MSDelay(10);
  }
```

```
    sprintf(transmitter_buffer,"AT+CWJAP=\"%s\",\"%s\"\r\n",ssid,password);
    UARTTransmitString(transmitter_buffer,strlen(transmitter_buffer));
    CleanTransmitterBuffer();
    MSDelay(2000);
    CleanReceiverBuffer();
}
```

To connect the server we pass ip and port number of the server to ConnectServer function. This function transmits AT commands to the ESP8266 by UART.

AT+CIPMUX sets the number of available TCP connections. AT+CIPSTART connects esp8266 to server

```
void ConnectServer(char *IP, char *port){

    UARTTransmitString("AT+CIPCLOSE\r\n",(unsigned short)strlen("AT+CIPCLOSE\r\n"));
    Enable_RX_Interrupt;
    MSDelay(1000);
    while(!(IsReceiveSendOK())){
        UARTTransmitString("AT+CIPCLOSE\r\n",(unsigned short)strlen("AT+CIPCLOSE\r\n"));
        Enable_RX_Interrupt;
        MSDelay(1000);
    }

    UARTTransmitString("AT+CIPMUX=0\r\n",(unsigned short)strlen("AT+CIPMUX=0\r\n"));
    Enable_RX_Interrupt;
    MSDelay(1000);
    while(!(IsReceiveSendOK())){
        UARTTransmitString("AT+CIPMUX=0\r\n",(unsigned short)strlen("AT+CIPMUX=0\r\n"));
        Enable_RX_Interrupt;
        MSDelay(1000);
    }

    unsigned short Length = sprintf(transmitter_buffer,"AT+CIPSTART=\"TCP\",\"%s\",%s\r\n",IP,pc
    UARTTransmitString(transmitter_buffer,Length);
    Enable_RX_Interrupt;
    MSDelay(2000);
    while(!(IsReceiveSendOK())){
        UARTTransmitString(transmitter_buffer,Length);
        Enable_RX_Interrupt;
        MSDelay(2000);
    }
    CleanTransmitterBuffer();
}
}
```

To connect the broker we pass client id, user name and user password to ConnectServer function. This function transmits AT commands to the ESP8266 by UART.

This function is responsible for preparing MQTT packages by calculating the necessary lengths and filling the buffers.

```
 void ConnectBroker(char *ClientID, char *UserName, char *UserPassword){

    uint16_t KeepAlive = 100;
    uint8_t  Connect = 0x10;
    uint8_t  Level = 0x04;
    uint8_t  Flag = 0xC2;
    uint16_t ProtocolNameLength = strlen(protocol_name);
    uint16_t ClientIDLength = strlen(ClientID);
    uint16_t UserNameLength = strlen(UserName);
    uint16_t UserPasswordLength = strlen(UserPassword);

    uint8_t RemainLength = 2 + ProtocolNameLength + 6 + ClientIDLength + 2 + UserNameLength + 2
    uint16_t TotalLength = sprintf(transmitter_buffer,"%c%c%c%c%s%c%c%c%c%c%s%c%c%s%c%c%s",(ch

    CleanTransmitterBuffer();
    sprintf(transmitter_buffer,"AT+CIPSEND=%d\r\n",TotalLength);
    UARTTransmitString(transmitter_buffer, (unsigned short)TotalLength);
    CleanTransmitterBuffer();
    MSDelay(100);
    CleanReceiverBuffer();

    sprintf(transmitter_buffer,"%c%c%c%c%s%c%c%c%c%c%s%c%c%s%c%c%s",(char)Connect,(char)Remain
    UARTTransmitString(transmitter_buffer, (unsigned short)TotalLength);
    CleanTransmitterBuffer();
    MSDelay(100);
    CleanReceiverBuffer();
}
```

3.3 LCD

LCD code checks each char in switch case to set pins high or low.

4 Conclusions

At the conclusion writing low level code and creating a stable system was a great adventure for us. Due to the time scarce, absence of third member of the group and knowledge level it was hard to build.

But we learned lot from our mistakes and now we know more about building stable system and choosing the right equipments and members.

All the code also published on github