Monty Sourjah

CPE301 – SPRING 2018

Design Assignment Midterm1 IOT Project

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS | Y |  |
| 2. | INITIAL CODE OF TASK 1/A | Y |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B | N/A |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C | N/A |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D | N/A |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E | N/A |  |
| 4. | SCHEMATICS | Y |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT | Y |  |
| 5. | SCREENSHOT OF EACH DEMO | Y |  |
| 6. | VIDEO LINKS OF EACH DEMO | Y |  |
| 7. | GOOGLECODE LINK OF THE DA | Y |  |
|  |  |  |  |
|  |  |  |  |

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

List of Components used

Atmega328P Xplained, External power supply(3.3V),LM34 temperature sensor,3Kohms resister,2.2Kohms resister,4.7Kohms resister, FTDI 3.3V connector.

Block diagram with pins used in the Atmega328P:

ESP8266-01

WiFi Module

GND GND

Atmega 328P Xplained Mini Board

LM34 Temperature Sensor

VCC VCC 5V VCC

Tx PD0

Rx PD1 PC0 Signal

EN GND GND

RST

4.7Kohms Resister

3Kohms Resister

2.2Kohms Resister

Power Supply

3.3 Volts

AT Command Block Diagram:

Tx Rx

PC Terminal Software(Arduino/Putty)

Power Supply 3.3V

FTDI Connector 3.3V

ESP8266

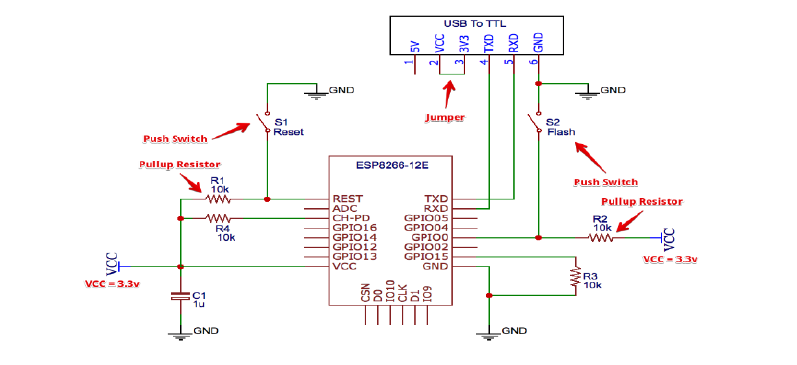
Rx Tx

GND

EN

VCC

**ESP 8266 Firmware Flashing Circuit:**





**2.INITIAL/DEVELOPED CODE OF TASK 1/A**

/\*

\* Midterm1\_IOT.c

\*

\*/

#define *F\_CPU* 16000000 //16MHz

#include <avr/io.h>

#include <avr/interrupt.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <util/delay.h>

#define FOSC 16000000UL

//#define BAUD 38400

//#define BAUD 115200

#define BAUD 9600

#define SETUP\_UBRR (FOSC/16)/BAUD-1

*uint16\_t* ubrr = SETUP\_UBRR;

char AT[] = "AT\r\n";

char ATRST[]="AT+RST\r\n";

char ATCIPSEND[]="AT+CIPSEND=%d\r\n";

//char ATCWJAP[] ="AT+CWJAP=\"SSID\",\"PASSWORD\"\r\n";

char CIPSTART[] = "AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n";

//char SEND\_DATA[] = "GET https://api.thingspeak.com/update?api\_key=THINGSPEAKSENDKEY&field1=%0.2f\r\n\r\n\r\n\r\n";

char CLOSE[]="AT+CIPCLOSE\r\n";

void UART\_init(void)

{

UBRR0H = (*uint8\_t*)(ubrr>>8);

UBRR0L = (*uint8\_t*)ubrr;

/\* Double speed off \*/

UCSR0A = 0 ;

/\* Enable receiver and transmitter \*/

UCSR0B=(1<<RXEN0)|(1<<TXEN0);

/\* Frame: 8 bit/ 1 stop bit / no parity\*/

UCSR0C=(1<<UCSZ00)|(1<<UCSZ01);

}

void UARTSendByte(char u8Data)

{

/\* wait while previous byte is completed \*/

while(!(UCSR0A&(1<<UDRE0))){};

/\* Transmit data \*/

UDR0=u8Data;

}

void USARTSendStr(char\* \_str)

{

int thesize=*strlen*(\_str);

for (*uint8\_t* i=0; i<thesize;i++)

{

UARTSendByte(\_str[i]);

}

}

void adc\_init(void)

{

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0); // // use AVcc as the reference

/\*

ADC operates within a frequency range between 50Kz and 200Kz

With a prescaler of 128 at CPU clock frequency of 16Mz, we will be in range:

F\_ADC=F\_CPU/128 = 125

So:

\*/

ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 128 pre-scale for 16Mhz

ADCSRB = 0; // 0 for free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

}

void sendTemperature(float temp)

{

char send\_data[128];

char atcipsend[32];

*int16\_t* len;

/\* Format data to send according to temperature. Returns length of string. \*/

len = *sprintf*(send\_data, SEND\_DATA, temp);

/\* Format AT+CIPSEND=%d according to data length to be sent \*/

*sprintf*(atcipsend, ATCIPSEND, len);

/\* Restart ESP8266 \*/

USARTSendStr(ATRST);

*\_delay\_ms*(2000);

/\* Connect to Access Point \*/

USARTSendStr(ATCWJAP);

*\_delay\_ms*(5000);

/\* Establish TCP connection \*/

USARTSendStr(CIPSTART);

*\_delay\_ms*(2000);

/\*Establish length of data will be sent \*/

USARTSendStr(atcipsend);

*\_delay\_ms*(2000);

/\* Send the data \*/

USARTSendStr(send\_data);

*\_delay\_ms*(2000);

/\* Close connection \*/

USARTSendStr(CLOSE);

}

int main(void)

{

*int16\_t* adc\_value;

float temp;

adc\_init();

UART\_init();

while (1)

{

/\* Start the ADC conversion \*/

ADCSRA |= (1 << ADSC);

/\* Wait for completion. ADSC reads 1 while still in progress \*/

while (ADCSRA & (1<<ADSC));

/\* save ADC value to a variable \*/

adc\_value=ADCL;

adc\_value = (ADCH<<8) + adc\_value;

/\*

For LM34:

Vout = + 10.0 mV/ºF

So the temperature will be:

temp = adc\_value \* (5V/adcresolution) / 10mV

or:

\*/

temp = ((float)adc\_value \* 5 / 1024) / 0.010;

sendTemperature(temp);

/\*

Delay 60 seconds (thingspeak.com restricts to 15 seconds delay between queries on the free account,

but it appears that sometimes it is much longer than that).

\*/

//\_delay\_ms(60000);

*\_delay\_ms*(15000);

}

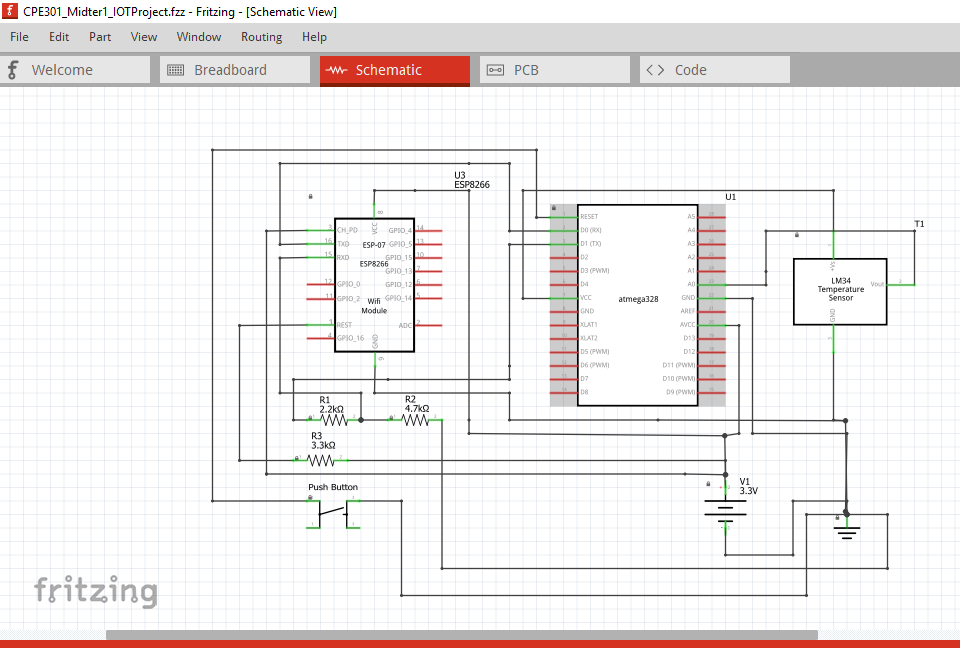
}

1. **MODIFIED CODE OF TASK 2/A from TASK 1/A**

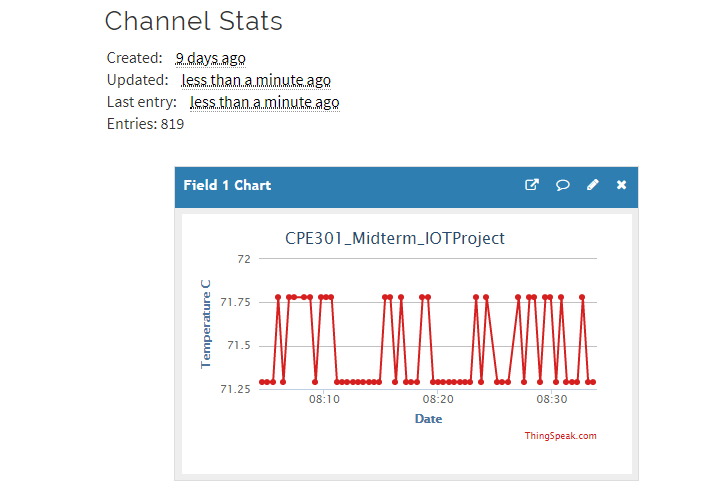
N/A

1. **SCHEMATICS**

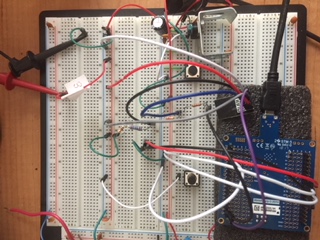
Use fritzing.org



**4.SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**



1. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



**5.VIDEO LINKS OF EACH DEMO**

<http://www.youtube.com/watch?v=MAIZdVF1_DU>

1. **GITHUB LINK OF THIS DA**

<https://github.com/ballasl/CPE301_Midterm1>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Monty Sourjah