Audie Escala

CPE301 – SPRING 2018

Midterm 1

**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 |  |  |
| 2. | INITIAL CODE OF TASK 1/A |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2 |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3-6 |  |  |
| 5. | THE ENTIRE C CODE. |  |  |
| 6. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 7. | YOUTUBE/GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

List of Components used

ATmega328P Xplained Mini

NodeMCU Wi-Fi ESP8266MOD

LM34 Temperature Sensor

1. **INITIAL/DEVELOPED CODE IN ASSEMBLY OF TASK 1**

; ==============================================================================

; TASK 1

;===============================================================================

; ADC reads the value from the LM34 temperature sensor

void adcGO()

{

char repeatTemp[13]= "Temperature: "; //header for temperature

char temperature[2] = "00"; //temperature string array

char temp; //temporary place holder

/\*

convert the read ADCvalue to temperature

500.0=>(Vref \* 100)=>(5V \* 100)

divide by 1024, the max for the ADC values (0-1024)

\*/

ADCvalue = ADC;

ADCvalue = (ADCvalue)\*(500.0/1024.0);

/\* converts value into ascii \*/

temp = (ADCvalue/10); //divide by 10 for tenths place digit

temperature[0] = temp + 48; //add ASCII '0'

temp= (ADCvalue%10); //modulo by 10 for ones place digit

temperature[1] = temp + 48; //add ASCII '0'

SEND\_DATA[44] = temperature[0]; // changes the sent data integer value

SEND\_DATA[45] = temperature[1]; // changes the sent data integer value

}

1. **DEVELOPED CODE IN ASSEMBLY OF TASK 2**

; ==============================================================================

; TASK 2

;===============================================================================

; Display the value to UART

void USARTsend(unsigned char Data) // function for sending data to the stream

{

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

UDR0=Data;

}

void send\_AT(unsigned char message[])

{

unsigned char i = 0;

while(message[i] != '\0')

{

USARTsend(message[i]); // sends data to esp

i++;

}

}

1. **DEVELOPED CODE IN ASSEMBLY OF TASK 3-6**

; ==============================================================================

; TASK 3 - 6

;===============================================================================

; Display temperature sensor value as a graph in Thing speak

int main(void)

{

unsigned char AT[] = "AT\r\n"; // check if OK

unsigned char CIPMUX[] = "AT+CIPMUX=1\r\n"; // set to multiple connections

unsigned char CIPSTART[] = "AT+CIPSTART=0,\"TCP\",\"api.thingspeak.com\",80\r\n"; // starts connection to thingspeak

unsigned char CIPSEND[] ="AT+CIPSEND=0,48\r\n"; // sends 48 bytes

unsigned char SEND\_DATA[] = "GET /update?api\_key=XJD2HOV04L2UOCX2&field1=70\r\n"; // send data to thingspeak

USARTinit(); //initialize USART

*\_delay\_ms*(500);

ADCinit(); //initialize the ADC

while(1)

{

adcGO(); // Get current ADC value

*\_delay\_ms*(200);

send\_AT(AT);

*\_delay\_ms*(1000);

send\_AT(CIPMUX);

*\_delay\_ms*(1000);

send\_AT(CIPSTART);

*\_delay\_ms*(1000);

send\_AT(CIPSEND);

*\_delay\_ms*(1000);

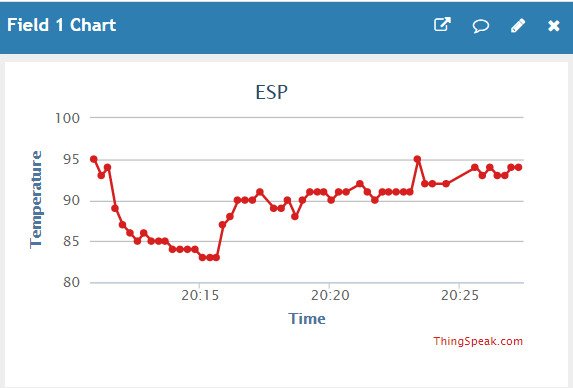
send\_AT(SEND\_DATA);

}

return 0;

}

1. **ENTIRE DEVELEPED CODE IN C**
2. #define *F\_CPU* 16000000UL
3. #define FOSC 16000000 // Clock Speed
4. #define BAUD 9600
5. #define MYUBRR FOSC/16/BAUD-1
6. #include <avr/io.h> //standard AVR header
7. #include <stdint.h> // need for uint8\_t
8. #include <util/delay.h> //delay header
9. #include <avr/interrupt.h>
10. void USARTinit (void)
11. {
12. /\* set baud rate \*/
13. UBRR0H = (MYUBRR >> 8); //high value of baud rate
14. UBRR0L = MYUBRR; // low value of baud rate
16. UCSR0B |= (1 << RXEN0) | (1<<TXEN0); //enable receiver and transmitter
17. UCSR0B |= (1 << RXCIE0); // enable receiver input
18. UCSR0C = ((1<<UCSZ01)|(1<<UCSZ00)); //asynchronous
20. }
21. volatile *uint8\_t* ADCvalue; // Global Variable of the returned value from ADC
22. void ADCinit(void) // function for Initializing ADC
23. {
24. ADMUX |= (1 << REFS0); //use AVcc as ref
25. ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // 128 Prescale for 16MHz
26. ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable
27. ADCSRB = 0; // Free running mode
28. ADCSRA |= (1 << ADEN); // Enable the ADC
29. ADCSRA |= (1 << ADIE); // Enable interrupts
30. ADCSRA |= (1 << ADSC); // start conversion
31. }
32. void USARTsend(unsigned char Data) // function for sending data to the stream
33. {
34. while (!(UCSR0A & (1<<UDRE0)));
35. UDR0=Data;
36. }
37. void send\_AT(unsigned char message[])
38. {
39. unsigned char i = 0;
40. while(message[i] != '\0')
41. {
42. USARTsend(message[i]); // sends data to esp
43. i++;
44. }
45. }
46. unsigned char SEND\_DATA[] = "GET /update?api\_key=XJD2HOV04L2UOCX2&field1=70\r\n"; // send data to thingspeak
47. void adcGO()
48. {
49. char repeatTemp[13]= "Temperature: "; //header for temperature
50. char temperature[2] = "00"; //temperature string array
51. char temp; //temporary place holder
53. /\*
54. convert the read ADCvalue to temperature
55. 500.0=>(Vref \* 100)=>(5V \* 100)
56. divide by 1024, the max for the ADC values (0-1024)
57. \*/
58. ADCvalue = ADC;
59. ADCvalue = (ADCvalue)\*(500.0/1024.0);
61. /\* converts value into ascii \*/
62. temp = (ADCvalue/10); //divide by 10 for tenths place digit
63. temperature[0] = temp + 48; //add ASCII '0'
64. temp= (ADCvalue%10); //modulo by 10 for ones place digit
65. temperature[1] = temp + 48; //add ASCII '0'
66. SEND\_DATA[44] = temperature[0]; // changes the send data integer value
67. SEND\_DATA[45] = temperature[1]; // changes the send data integer value
68. }
69. int main(void)
70. {
71. unsigned char AT[] = "AT\r\n"; // check if OK
72. unsigned char CIPMUX[] = "AT+CIPMUX=1\r\n"; // set to multiple connections
73. unsigned char CIPSTART[] = "AT+CIPSTART=0,\"TCP\",\"api.thingspeak.com\",80\r\n"; // starts connection to thingspeak
74. unsigned char CIPSEND[] ="AT+CIPSEND=0,48\r\n"; // sends 48 bytes
76. USARTinit(); //initialize USART
77. *\_delay\_ms*(500);
78. ADCinit(); //initialize the ADC
79. while(1)
80. {
81. adcGO(); // Get current ADC value
82. *\_delay\_ms*(200);
83. send\_AT(AT);
84. *\_delay\_ms*(1000);
85. send\_AT(CIPMUX);
87. *\_delay\_ms*(1000);
88. send\_AT(CIPSTART);
89. *\_delay\_ms*(1000);
90. send\_AT(CIPSEND);
91. *\_delay\_ms*(1000);
92. send\_AT(SEND\_DATA);
93. }
94. return 0;
95. }
96. **TASK 5 SCREEN CAPTURE**



1. **Flowchart**
2. **YOUTUBE & GITHUB LINK OF DA1**

https://www.youtube.com/watch?v=0RT58ccR8AY

https://github.com/escalaa/Midterm

**Student Academic Misconduct Policy**

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

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

Audie Escala