CPE301 – SPRING 2019

MIDTERM I

Student Name: Billy Maddex

Student #: 2000928390

Student Email: [maddex@unlv.nevada.edu](mailto:maddex@unlv.nevada.edu)

Primary Github address: [maddex@unlv.nevada.edu](mailto:maddex@unlv.nevada.edu)

Directory: Midterm

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/Midterm, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

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

Atmega328PB-XMINI

LM35 Temperature Sensor → PINC5

ESP01 WiFi Module → PIND0/PIND1

1. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1**

/\*

\* Midterm.c

\*

\* Created: 10/27/2019 12:50:46 PM

\* Author : Billy

\*/

// 16MHz CPU clock

#define F\_CPU 16000000UL

// 115200 baud rate setting

#define URRR\_115200 8

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/atomic.h>

#include <stdlib.h>

#include <stdio.h>

#include <util/delay.h>

// 15s timer variable, interval is 14 because we count from 0

const int INTERVAL = 14;

int transmit\_time;

// terminal setup and output function declarations

void uart\_init(unsigned int ubrr);

void uart\_send\_string(char \*data);

// ADC function declarations

void adc\_init(void);

// ESP01 setup declaration

void esp01\_init(void);

// function to transmit temperature value

void xmit\_temp(float temp);

// TIMER1 CTC interrupt handler

ISR(TIMER1\_COMPA\_vect)

{

// when transmit time is up, do something

if (transmit\_time-- == 0)

{

// start ADC conversion

ADCSRA |= (1 << ADSC);

// reset transmit\_time

transmit\_time = INTERVAL;

}

}

ISR(ADC\_vect)

{

// collect and compute ADC value

// value stored in ADC

float cel = ((5.0 \* ADC) / 1000.0) + 4;

float fahr = (1.8 \* cel) + 32;

// transmit the temp data

xmit\_temp(fahr);

}

int main(void)

{

// configure terminal

uart\_init(URRR\_115200);

// configure and start ADC

adc\_init();

// configure ESP01 and connect to WiFi

esp01\_init();

// initialize transmit\_time

transmit\_time = INTERVAL;

// configure interrupt handling

// enable TIMER1 COMPA interrupt

TIMSK1 |= (1 << OCIE1A);

// set TIMER1 to CTC operations

TCCR1B |= (1 << WGM12);

// TIMER1 COMPA = X

OCR1A = 15620;

// start TIMER1 with prescalar = 1024

TCCR1B |= (1 << CS12) | (1 << CS10);

// enable global interrupt

sei();

// loop forever

while (1);

}

// configure terminal output

void uart\_init(unsigned int ubrr)

{

// set baud rate

UBRR0H = (unsigned char)(ubrr>>8);

UBRR0L = (unsigned char)ubrr;

// set terminal to input and output

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

// set output mode 8-bit data

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

UCSR0C =

// async operation

(0 << UMSEL01) |

(0 << UMSEL00) |

// Parity disabled

(0 << UPM01) |

(0 << UPM00) |

// stop Bits = 1bit

(0 << USBS0) |

// 8 Data bits

(1 << UCSZ01) |

(1 << UCSZ00) |

// for Synch Mode only clock polarity

(0 << UCPOL0);

}

void uart\_send\_string(char \*data)

{

// send until the string is done

while((\*data != '\0'))

{

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

UDR0 = \*data;

data++;

}

// required delay between UART commands

\_delay\_ms(1800);

}

// configure ADC input

void adc\_init (void)

{

// set PORTC to input for ADC

DDRC = 0x00;

ADMUX =

// Reference Selection Bits

// AVcc with external capacitor at AREF

(0 << REFS1) |

(1 << REFS0) |

// ADC Left Adjust Result

(1 << ADLAR) |

// Analog Channel Selection Bits

// ADC5 (PC5 PIN28)

(0 << MUX3) |

(1 << MUX2) |

(0 << MUX1) |

(1 << MUX0);

ADCSRA =

// ADC ENable

(1 << ADEN) |

// ADC Start Conversion

(0 << ADSC) |

// ADC Auto Trigger Enable

(0 << ADATE) |

// ADC Interrupt Flag

(0 << ADIF) |

// ADC Interrupt Enable

(1 << ADIE) |

// ADC Prescaler Select Bits

// prescalar = 128

(1 << ADPS2) |

(1 << ADPS1) |

(1 << ADPS0);

}

void xmit\_temp(float temp)

{

// output strings

char send[20];

char report[80];

int report\_len = 0;

// generate output string

snprintf(report, sizeof(report), "GET /update?api\_key=TNTAKRCSHOJW0ANL&field1=%.2f\r\n\r\n", temp);

// get number of bytes to send

while (report[report\_len] != '\0')

{

report\_len++;

}

// generate SEND AT string based on byte size of report

snprintf(send, sizeof(send), "AT+CIPSEND=%d\r\n", report\_len);

// output to terminal

ATOMIC\_BLOCK(ATOMIC\_FORCEON)

{

uart\_send\_string("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r\n");

uart\_send\_string(send);

uart\_send\_string(report);

}

}

// configure ESP01 and connect to WiFi

void esp01\_init(void)

{

uart\_send\_string("AT\r\n");

uart\_send\_string("AT+CWMODE=1\r\n");

uart\_send\_string("AT+CIPSTA=\"10.7.7.22\"\r\n");

uart\_send\_string("AT+CWJAP=\"xxxxxxxx\",\"xxxxxxxx\"\r\n");

uart\_send\_string("AT+CIPMUX=0\r\n");

}

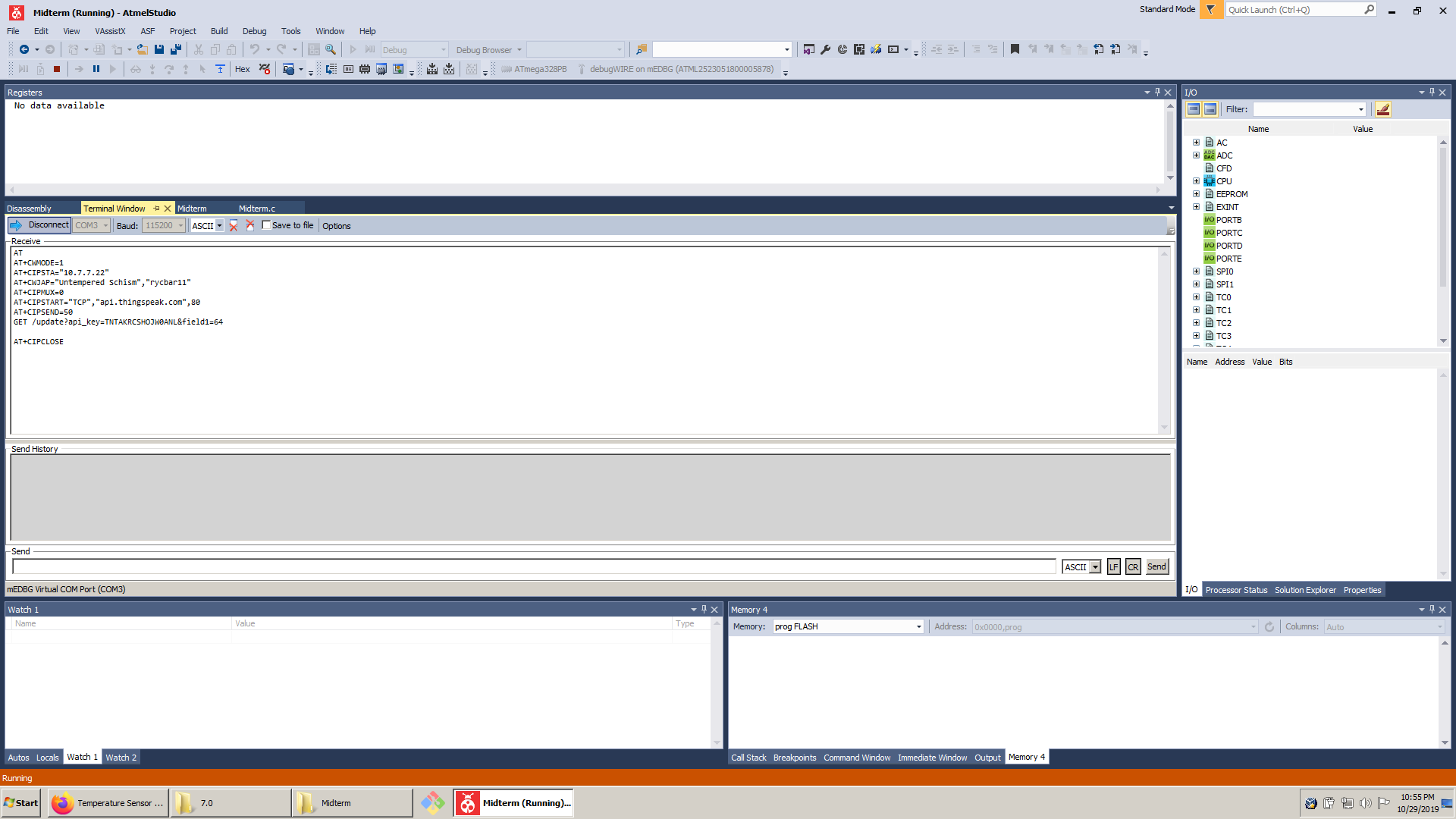
1. **DEVELOPED MODIFIED CODE OF TASK 2 from TASK 1**

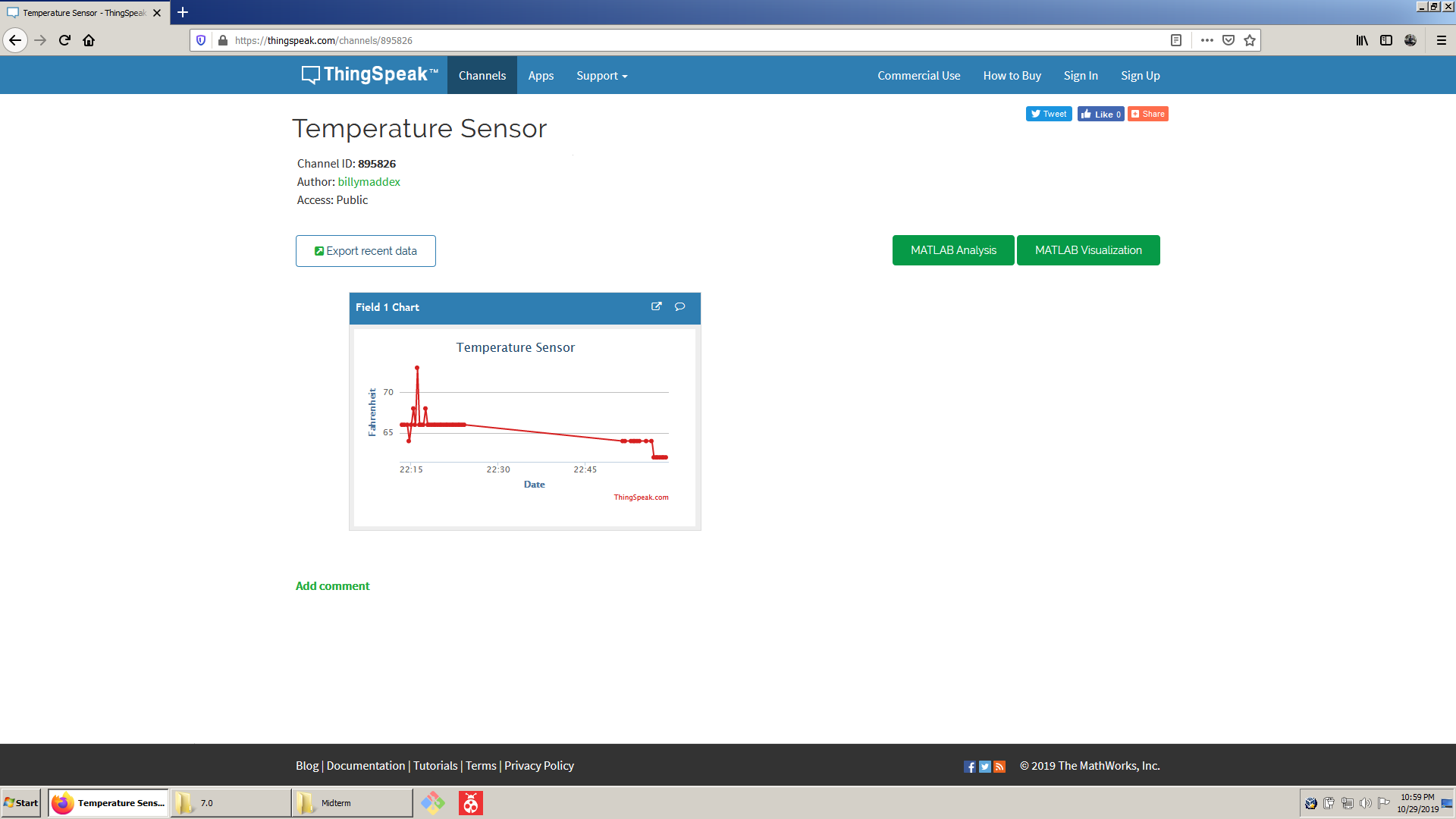
// no modified code

1. **SCHEMATICS**

Use fritzing.org

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





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



1. **VIDEO LINKS OF EACH DEMO**

https://youtu.be/upMDbBFYtDk

https://thingspeak.com/channels/895826

1. **GITHUB LINK OF THIS DA**

<https://github.com/billymaddex/fluffy-chainsaw/tree/master/Midterm>

**Student Academic Misconduct Policy**

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

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

Billy Maddex