Title: Testing the temperature

Goal:

* Read LM34 temperature sensor with Atmega328p over SPI
* Transfer data to ESP8266 module through UART
* Push reading onto Thingspeak.com account
* Create graph comparing current temperature to forecasted temperature

Deliverables:

The main deliverable of this assignment is a graph plotting both current room temperature as well as a forecasted temperature.

Literature Survey:

Having the ability to plot the current temperature over time from a sensor can help in maintaining a desired temperature. By integrating this device with a homes heating system it could accomplish that goal.

Components:

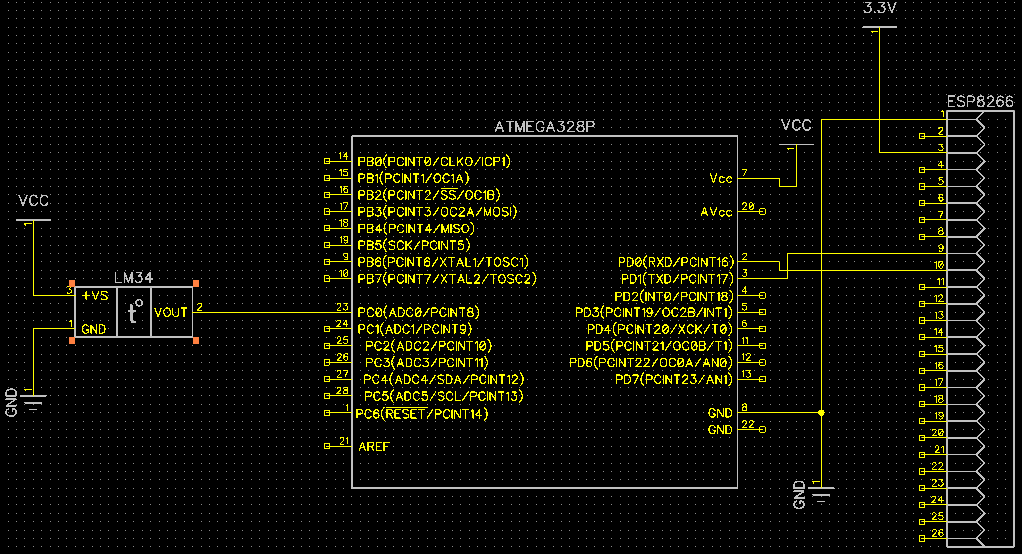
Atmega328p –Microcontroller (SPI, UART interface)

LM34 - Temperature Sensor

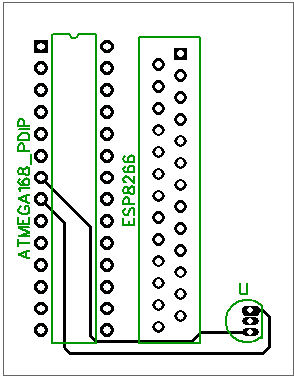
ESP8266 – WiFi module (3.3v)

USB FTDI – 3.3 V adapter

USB cable – 5 V power

Schematics: 

Initial PCB:



Implementation:

* ATMega328P connected through SPI to Temperature sensor
* ATMega328P connected through UART to ESP8266
* Every 15 minutes ESP8266 pushes temperature data to web server (Thingspeak.com)
* Web server compares current temperature read with forecasted value

Snapshots/Screenshots:

CODE: (with comments)

//project is still not working, attempting to get test.c to work //then move on to interfacing sensor with atmega with //ESP8266.

/\*

\* test.c

\* Author: battled

\*/

/\*

NOTE ESP8266 MODULE AUTOMATICALLY CONFIGURED TO CONNECT TO

AT+CWJAP="Shabrya's iPhone","lott1993"

TEST CODE TO ATTEMPT TO PUSH INTEGER 42 TO THE MY CLOUD

\*/

#define THING\_SPEAK\_IP\_STR "184.106.153.149" /\* thingspeak.com IP Address \*/

#define THINK\_SPEAK\_IP\_PORT 80 /\* port number \*/

#define THING\_SPEAK\_KEY\_STR "Z4SEGG9JLKWHAE8D" /\*My API key \*/

#define THING\_SPEAK\_CHANNEL 20696 /\* channel ID \*/

#define THING\_SPEAK\_LABEL\_STR "field1"

#define MOSI 4 //mosi is pb3

#define SCK 6 //SCK is pb5

#define SS 3 //SS is pb2

#define F\_CPU 8000000UL //8 MHz xtal crystal

#include <avr/io.h> //

#include <avr/interrupt.h>

#include <util/delay.h>

#include <string.h> //for string concatenation

#include <stdio.h> //for sprintf

//

#define MOSI 4 //mosi is pb3

#define SCK 6 //SCK is pb5

#define SS 3 //SS is pb2

#define F\_CPU 8000000UL //8 MHz xtal crystal

//static int i = 0; //used to collect and then average temperature and count interrupts

void SPI\_transmit(char\* cData)

{

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

{

while(!(SPSR & (1<<SPIF)))

SPDR = \*cData;

cData++;

}

}

void SPI\_init(void)

{

DDRB = ((1<<SS)|(1<<MOSI)|(1<<SCK)); //spi pins on port b MOSI SCK,SS outputs

SPCR = ((1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<CPOL)|(1<<CPHA)); // SPI enable, Master, f/16

SPI\_transmit("AT+CIPMUX=1\n\r"); //one time initialization of CIPMUX

}

int main(void)

{

DDRC &= ~(1<<PORTC0); //PC0 is analog input

SPI\_init();

while(1)

{

SPI\_transmit("AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80\n\r");

\_delay\_ms(1000);

SPI\_transmit("AT+CIPSEND=4,50\n\r");

\_delay\_ms(1000);

SPI\_transmit("GET /update?key=Z4SEGG9JLKWHAE8D&field1=42\n\r");

\_delay\_ms(1000);

SPI\_transmit("GET /update?key=Z4SEGG9JLKWHAE8D&field1=42\n\r");

\_delay\_ms(1000);

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\*

\*

\* Final.c

\* Author: battled

\*Code to read from LM-34 sensor, covert the temp and transmit it to ESP8266

\*/

#define MOSI 4 //mosi is pb3

#define SCK 6 //SCK is pb5

#define SS 3 //SS is pb2

#define THING\_SPEAK\_IP\_STR "184.106.153.149" /\* thingspeak.com IP Address \*/

#define THINK\_SPEAK\_IP\_PORT 80 /\* port number \*/

#define THING\_SPEAK\_KEY\_STR "Z4SEGG9JLKWHAE8D" /\*My API key \*/

#define THING\_SPEAK\_CHANNEL 20696 /\* channel ID \*/

#define THING\_SPEAK\_LABEL\_STR "field1"

#define F\_CPU 8000000UL //8 MHz xtal crystal

#include <avr/io.h> //

#include <avr/interrupt.h>

#include <string.h> //for string concatenation

#include <stdio.h> //for sprintf

#include <util/delay.h>

//

void usart\_init (void)

{

UCSR0B = (1<<TXEN0); //transmit enable

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

UBRR0L = 0x33; //baud rate 9600 at 8 MHz for UART read

}

void SPI\_tx\_string(char \*data) //takes a string and sends it serially

{

while(\*data != '\0'){ //send chars until NULL is found

while(!(SPSR & (1<<SPIF))); //wait for SPSR to be 1

SPDR = \*data; //push current char into spdr reg

data++; //point to next char

}

}

void SPI\_prepSend(void)

{

SPI\_tx\_string("AT+CIPSTART=4,\"TCP\",\"184.106.153.149\",80\n\r");

\_delay\_ms(1000);

SPI\_tx\_string("AT+CIPSEND=4,50\n\r");

\_delay\_ms(1000);

}

void SPI\_init(void)

{

DDRB = ((1<<SS)|(1<<MOSI)|(1<<SCK)); //spi pins on port b MOSI SCK,SS outputs

SPCR = ((1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<CPOL)|(1<<CPHA)); // SPI enable, Master, f/16

SPI\_tx\_string("AT+CIPMUX=1\n\r"); //one time initialization of CIPMUX

}

int main(void)

{

DDRC &= ~(1<<PORTC0); //PC0 is analog input

SPI\_init(); //initialize SPI

usart\_init(); //initialize USART

ADCSRA = ((1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0));// ADC prescaler 128 ADEN

ADMUX = ((1<<REFS1)|(1 << REFS0)); // select internal 1.1 V Ref w/ ext cap at AREF pin and ADC0 (default)

sei(); //enable interrupts

//configure timer 1 to interrupt every second

TCNT1 = 65536 - ((double)F\_CPU/256); // set timer to overflow in 1 sec

TCCR1A = 0; //normal mode

TCCR1B = 4; //prescaler = 256

TIMSK1 |= (1<<TOIE1); //enable interrupt on overflow timer 1

while(1); //wait for interrupts

return 0;

}

ISR(TIMER1\_OVF\_vect) //timer1 overflow ISR

{

TCCR1B = 0; //stop timer 1

TIFR1 = 1; //clear overflow flag

int adc\_temp; //stores ADC

float adc\_tempf; //float for calculations

int adc\_tempi; //integer part

char TmpTemp[44];

char daTemp[5];

//read ADC

ADCSRA |= (1<<ADSC); //start conversion

while((ADCSRA &(1<<ADIF)) == 0); //wait for conversion to finish

adc\_temp = ADC; //save ADC value

adc\_tempf = (float)adc\_temp \* (1.1 / 1024) / 0.01; //(ADC\*res/.01) (lm34 sf = 10mv/degF)

adc\_tempi = (int)adc\_tempf; //integer part

SPI\_prepSend();//prepare to upload the converted string

sprintf(TmpTemp,"GET /update?key=Z4SEGG9JLKWHAE8D&field1=");

//sprintf here is used to "print" to TempInt string and then dot is added at end

sprintf(daTemp,"%d\r\n",adc\_tempi);

strcat(TmpTemp, daTemp);

SPI\_tx\_string(TmpTemp);

SPI\_tx\_string(TmpTemp);

//output degrees Fahrenheit to cloud

//reset time 1 for interrupt

TCNT1 = 65536 - ((double)F\_CPU/256); //overflow in 1 sec

TCCR1A = 0; //normal mode

TCCR1B = 4; //prescaler = 256

return;

}

Reference:

<http://nodemcu.readthedocs.io/en/dev/en/upload/>

<https://github.com/nodemcu/nodemcu-firmware>

<https://www.mathworks.com/help/thingspeak/examples.html>

<https://sites.google.com/a/unlv.edu/unlvcpe301/projects>