ECE 4220

An Embedded System for the Optimization, Monitoring, Control, and Automation of a Green House Environment

Barrett Lamb

14112240

5/13/2015



Contents

[Table of Figures 2](#_Toc419473508)

[Abstract: 3](#_Toc419473509)

[Introduction: 4](#_Toc419473510)

[Background: 6](#_Toc419473511)

[Implementation: 8](#_Toc419473512)

[Experiments and Results: 11](#_Toc419473513)

[Discussion 14](#_Toc419473514)

[Appendices: 16](#_Toc419473515)

[Code: 16](#_Toc419473516)

[Works Cited: 24](#_Toc419473517)

# Table of Figures

[Figure 1: Tractor Sensors http://www.mdpi.com/1424-8220/13/8/10177/htm 6](#_Toc419473456)

[Figure 2 http://www.sunjournal.com/files/imagecache/story\_large/2014/04/26/CITtinkertecharduinoponics042814\_arduino.jpg Arduino greenhouse controller 6](#_Toc419473457)

[Figure 3 Professional system from Argus http://www.acs-automation.org/old\_site/images/pg67.jpg 7](#_Toc419473458)

[Figure 4 Hardware Implementation 9](#_Toc419473459)

[Figure 5 Software Implementation 10](#_Toc419473460)

[Figure 6 adc\_7250 test program 11](#_Toc419473461)

[Figure 7 Main program (results in video) 12](#_Toc419473462)

# Abstract:

The design for An embedded System for the Optimization, Monitoring, Control, and Automation of a Green House Environment does exactly what its long title spells outs. It is a system which is designed primarily for the automation of an environmental chamber which is at least somewhat controlled. The actual implementation focus’s heavily on the controlled automation of a fertilizer dosing tank of feed water to be applied to plants. The system ensures that the water level does not drop too low which could result in pump over heating or failure as well as preventing the tank from overflowing. The system also incorporates a dosing pump which can be set to a defined level for increasing or decreasing nutrient concentration.

The environment was assumed to be somewhat controlled though temperature and light data is taken and printed to a terminal screen for verification. The system does provide some amount of environmental control in that it will turn on axially lighting if the light in the room is sensed as too low to ensure that users could safely exit the area as well as to maintain even plant flowering and growth cycles. The system is designed as an add-on to an existing pumping/watering system.

# Introduction:

Recently, a computer science alumni, by the name of Mike Olson gave a talk called “The Engineer’s Century”, at the University of California – Berkeley. The talk primarily focused on the progression of man throughout history and particularly their utilization of resources. There was a graph which showed a relative number of resources that the average population would have at that time which remained mostly unchanged for the majority of human existence, occasionally increasing in times of prosperity and dropping during times like the Black Plague. At the end of the graph the line took an exponential curve up and shot off of the page. He asked the audience if they knew what that huge change that occurred in the 1800’s represented: the industrial revolution.

Man began using machines to do work that previously would have been unthinkable by one person or such a small few. Inventions like the steel plow and other farm implements massively increased the amount of land a farmer could grow on. These incredible mechanical machines made work easier, faster, and more efficient. There was one particular innovation that would change the mankind and the face of the planet forever, steam power.

Now, not only could man make machines that helped with intense physical labor, man could make machines that could complete physical labor powered by themselves with no exertion from an operator. Tractors, trucks, and trains were created all with the ability to create, lift, and move goods that were larger and could be moved faster and farther than any other time in human history. It truly was the age of the mechanical engineer.

At the end of the 1800’s into the 1900’s things really started to get exciting as experiments with electricity were done and a power grid established. Power could be transmitted across vast distances to turn motors, power lights, and eventually computers. Now enters the time of the electrical and computer engineers. We now not only plow our fields with tractors but we do so with tractors outfitted with extremely accurate GPS coordinates sent from satellites in space that are logged from electrical sensors to measure things like Ph, electrical conductivity, soil grade, moisture level, etc… By taking this information, companies like Monsanto can pick specifically genetically modified seeds which are ideal for those conditions and improve yields of the same land by as high as thirty percent.

Thirty percent more food can be produced almost solely due to a system which has been implemented which can use electrical sensors and then perform meaningful analysis of large amounts of data to produce as close to ideal conditions as possible. This was the main motivation behind An Embedded System for the Optimization, Monitoring, Control, and Automation of a Green House Environment. Certain aspects of the project were implemented in fair detail including the control and automation, where the optimization and monitoring could be expanded. This was the main goal as it seemed appropriate that all the control and sensing of actual inputs should be done locally on an embedded system to ensure reliability but data processing and optimization could be accomplished with more powerful hardware remotely with the ability to transfer back new values such as the dosing time of the fertilizer pump or the light level requiring axillary lighting.

# Background:

As stated a number of companies and individuals use similar technology every day. Monsanto has contracted DuPont to implement these arrays of tractor sensors on current customers.



Figure : Tractor Sensors http://www.mdpi.com/1424-8220/13/8/10177/htm

A quick search on the web will show that there are endless solutions for green house environmental control and automation varying greatly in degree of complexity, level of control, interface, and price. Some are systems based on popular in expensive microcontrollers with simple feedback loops

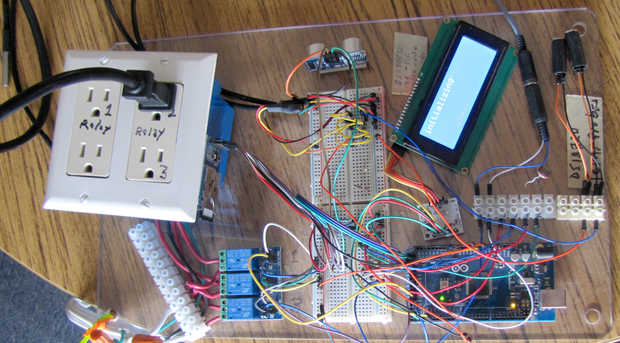


Figure http://www.sunjournal.com/files/imagecache/story\_large/2014/04/26/CITtinkertecharduinoponics042814\_arduino.jpg Arduino greenhouse controller

Others are professionally designed systems with the support to control massive amounts of plants, water, lights and other physical connections with ability to sense, optimize, and control in a very precise manor.



Figure Professional system from Argus http://www.acs-automation.org/old\_site/images/pg67.jpg

Obviously, there is already a strong industry in place and would be difficult to revolutionize or even compete with professional designs so It seemed appropriate the system focus primarily on the preparation and control of nutrient/fertilizer solution in a mixing tank.

# Implementation:

As stated before, seeing as creating a full environmental control system seemed impractical and beyond the scope of this project the fine control of a nutrient mixing reservoir was the primary focus. This was accomplished using a number of pieces of hardware and software.

The hardware has a number of control and sensing elements. The primary sensing elements are a photocell based light sensing circuit, a thermistor temperature sensing circuit as well as two float sensors. The float sensors were implemented using simple memory mapped I/O from the TS-7250 port B digital I/O pins. Float 1 represented the lower float level which when low signaled water needed to be added and a higher float, float 2 which represented the tank being full. These inputs were measured in a simple polling method which certainly left room for improvement.

The temperature and light sensors were similar to each other in that they were both designed as 10k nominal voltage divider circuits which could be sensed using the TS-7250 EP9302 analog to digital converter as well as three header files and an example program for performing the actual conversion from Technologic Systems FTP repository site <ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7250-linux>. Code was modified but borrowed heavy amounts of its implementation from the example code as we discussed the operation and theory behind analog to digital converters but did little into actual implementation.

With these sensors water levels could be measured and temperature and light levels could be monitored and displayed. This led to the control portion of the hardware which was in large an 8-relay board which could be triggered through the TS-7250 digital I/O pins by masking the bit corresponding to the desired relay 0 (active LOW configuration). Three relays were utilized in the final implementation: one for turning on 120 v power if axillary lighting was required, a 12 V peristaltic pump which could pump fertilizer solution for a set amount of time, and a 12 V solenoid valve which would open to fill the tank and close to stop from overflowing.



Figure Hardware Implementation

The software was implemented using one child process created with the fork command to convert and print temperature data (it had no effect on control system) and another main parent process which ran all other sensing, conversion, and control. The sensing of the analog temperature sensor was actually accomplished in the parent process where it would also convert the analog value into a voltage which could be passed through a pipe to the child process to be converted and then printed.

The main process was in charge of mapping all the appropriate registers, calibrating the analog channels, masking and reading the digital values, output masking relays, and converting other signals. In this case it could be said that the sensing was done in largely a polling fashion as the parent process would check all its inputs, make changes, wait, and do it again. Its response time was greatly degraded due to its sleep function making it possible to read and debug the system.



Figure Software Implementation

# Experiments and Results:

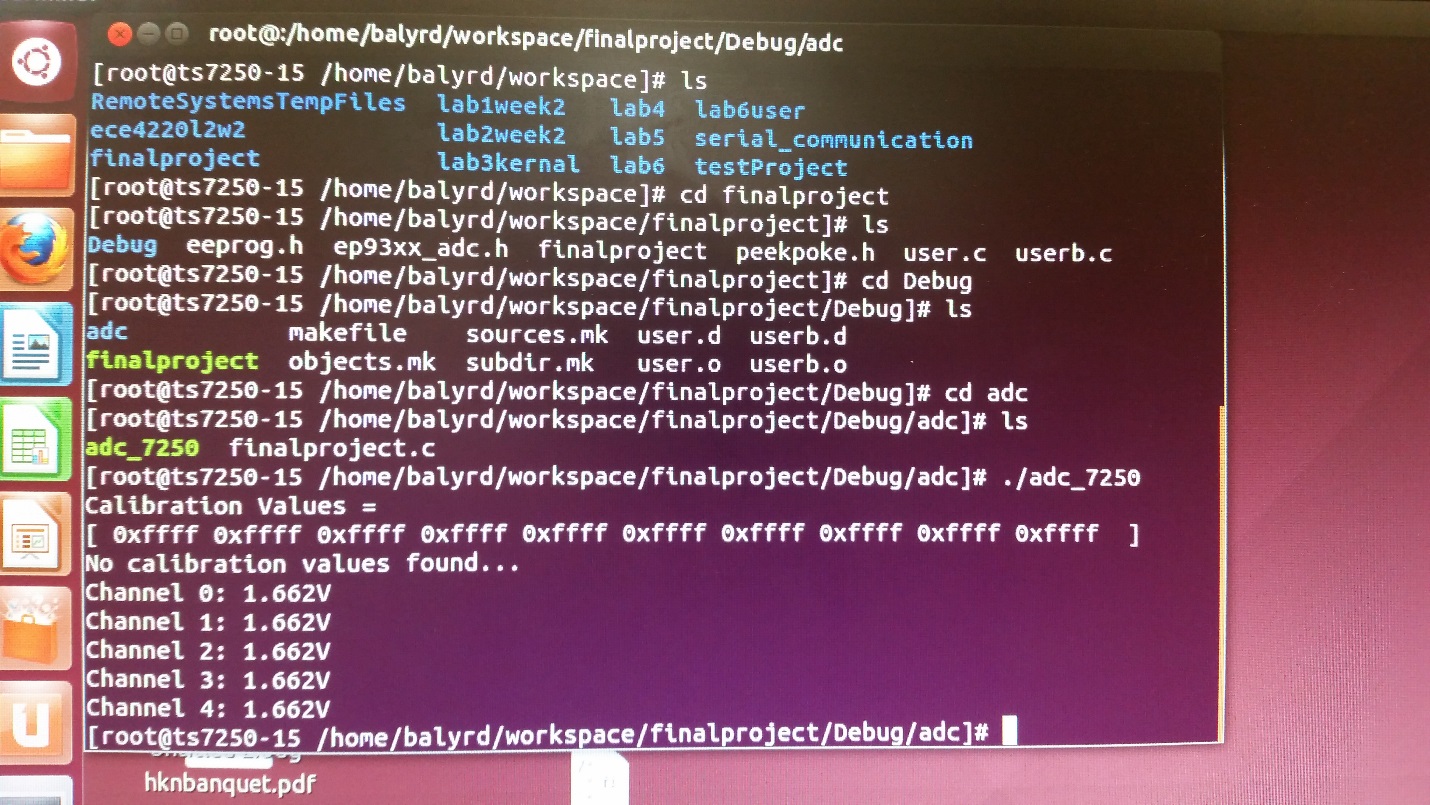
A number of experiments were performed to confirm the proper operation of the system and many could easily be tested individually of each other. The first step in testing the system once the hardware was thought to be set up correctly was reading the analog values from the temperature and light sensors. This could be done using a stock program from Technologic Systems called adc\_7250 available at their repository which would simply make analog readings of all its inputs. This meant that a reading of the light could be taken, then the light covered and another taken where the user should be able to observe a change if functioning properly. Seeing a displayed voltage of exactly half the source 3.3 V (1.662 V) was also a telltale that the circuit was not powered correctly. 

Figure adc\_7250 test program

A similar method was utilized to test the temperature sensor. After the analog readings were confirmed to be working the main program could be used to start taking measurements. In this setup if a user covered the photocell with their hand they would see the relay controlling the outlet for the axillary lighting turn on and then back off when removed. The temperature data was also printed as a voltage and in degrees Celsius repeatedly to ensure proper operation.

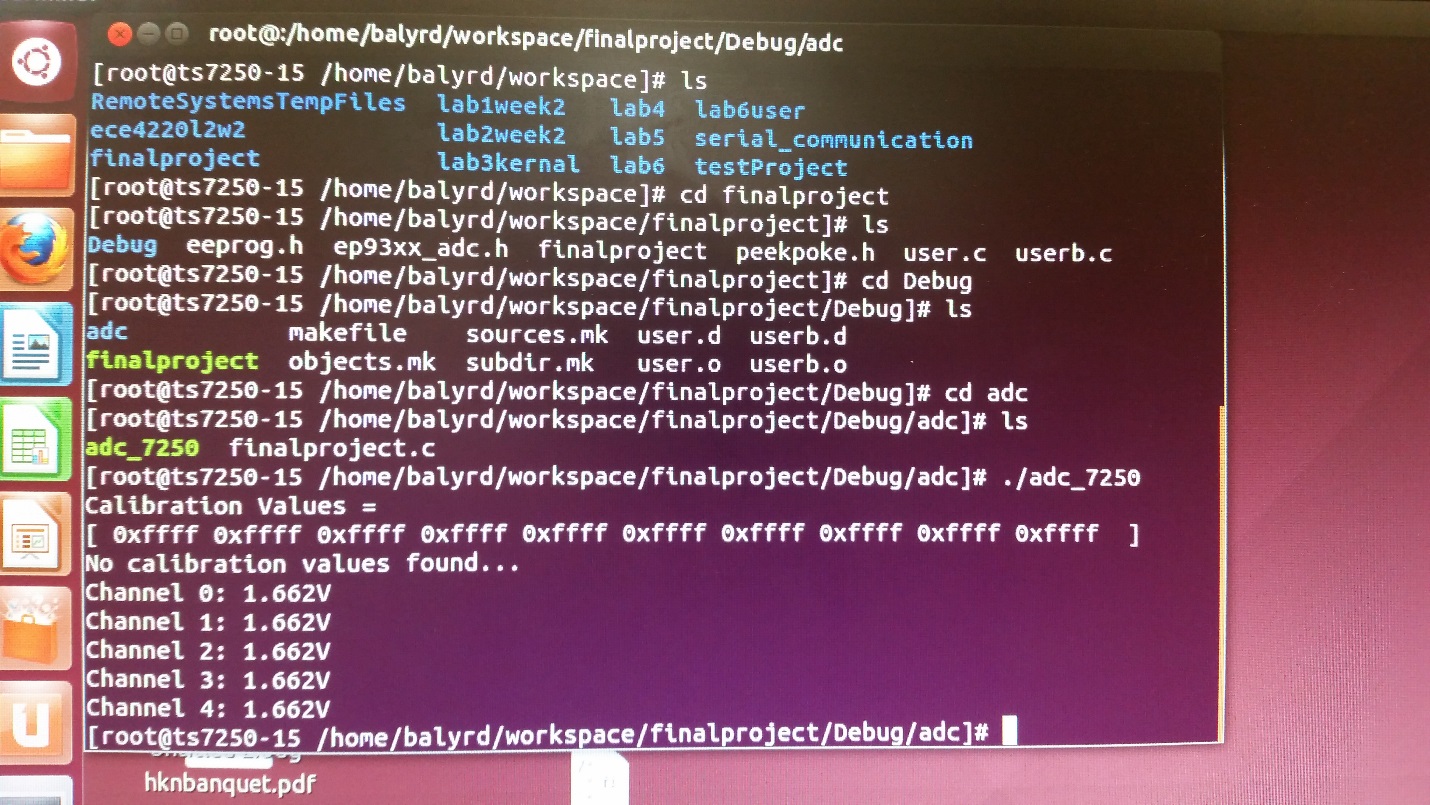


Figure Main program (results in video)

The operation of the digital input float sensors could easily be tested as well using the main program. As the tub was never actually filled with water both floats would start as logical LOW values. This would mean that the relay controlling the solenoid for filling the tub with water would be on as soon as you began running the program. The user would then raise the lower float, then the higher float and the relay controlling the solenoid would turn off while the relay controlling the dosing pump would turn on for the defined amount of time and then turn off. At that point lowering and raising the upper float would have no effect on the system. When the lower float was lowered the system would think it was empty again and the filling and dosing process could begin again. The system was also able to handle a faulty low level float as the solenoid would be turned off regardless of the lower float state and not overflow though would not protect against a faulty high where the system could end up running itself dry.

# Discussion

Overall the system seemed to work relatively well and was generally successful. Of course numerous improvements could have been made but the general control system worked as expected. The relays turned on at the correct times and analog readings could be converted and manipulated as well as transmitted or printed. One major concept that had originally been planned as an implementation was to have the float signals read as interrupts. This would mean that the water level could be controlled in real time and that it would not have to wait for the rest of the process to complete or the wait at the end of the loop which was associated with convenient reading for the user. The entirety of the printing probably should have been done in the child process including the light where it could be piped from the main so that main could continue running its sensing and control methods at a much higher frequency.

The largest piece of the system that was missing was the optimization software. As of current the child process just prints data to the terminal. It would be much more useful for the child process to send that data to a server somewhere where it could be indexed and processed to send back changes to timings and thresholds for example the dosing pump and the light requirements. Currently the dosing time is a single time defined at the top of the program although a more ideal situation would be a system which could be sent multiple dosing times for different types of fertilizers to truly test and optimize the system with multiple pumps.

Another piece which was intended to be implemented was an electrical conductivity meter which would work on a similar analog scale so that nutrient concentrations could be measured and recorded. This was not implemented primarily due to the sensor needing to be submerged in water to take readings which was not a good combination for prototyping with electronics.

The largest problems certainly seemed to come in relation to the analog signals. When trying to initially implement the analog reading was very unsure of how to accomplish this task. There was also a problem where a wire had been knocked out when the project was assumed to be finished as well and the ADC values would do nothing but print default values. It would constantly beg to be picked up another day but because the hardware scope was so large and entangled it was extremely difficult to put together and take apart the entire system each day so tests had to be done in a large uninterrupted time span which is not at all easy for a public lab at a university.

# Appendices:

## Code:

/\*

\* finalproject.c

\*

\* Created on: May 14, 2015

\* Author: balyrd

\*/

#include <unistd.h>

#include <sys/types.h>

#include <sys/mman.h>

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <assert.h>

#include "peekpoke.h" //header files available from manufacturer

#include "eeprog.h"

#include "ep93xx\_adc.h"

#define DATA\_PAGE 0x80840000

#define CALIB\_LOC 2027 //location of calibration values

#define DOSETIME 50000 //defined value for time to keep does pump on in us

/\* Prototypes \*/

static double read\_ADC\_channel(int channel); //take analog reading

static void init\_adc\_channel\_calibration(); //set up analog channel

int getLightLevel(double voltage); //converts light voltage to a level 1-5

/\* globals \*/

static unsigned long dr\_page, adc\_page, syscon\_page, pld\_page;

int stored\_cal[5][2]; //stored calibration values for ADC, first dimension are

// 5 channel, second dimension are 2 calibration values

int virgin = TRUE; //calibration detected?

//child process converts and prints temperature

void child(int pipe\_p,int pipe\_c){

double temp,temp\_v;

while(1)

{

read(pipe\_c,&temp\_v,sizeof(&temp\_v)); //read from pipe

temp=25\*(temp\_v/1.65);//temp in c

printf("temp: %f ,%.2f\n",temp\_v,temp); //print value

usleep(2000000);

}

}

int main(void)

{

double light\_v,temp\_v; //light and temp as voltage

int light,float1,float2,flag=0; //levels of floats and light

unsigned long \*pbdr, \*pbddr, \*fptr;//register pointers

unsigned long \*input;

int pid;

//create pipes one probably would have worked

int pipe\_p[2],pipe\_c[2];

pipe(pipe\_p);

pipe(pipe\_c);

pid=fork();

//setup child

if(pid==0) //child

{

close(pipe\_p[0]);

close(pipe\_c[1]);

child(pipe\_p[1],pipe\_c[0]);

}

//proper register memory mappings

int devmem = open("/dev/mem", O\_RDWR|O\_SYNC);

assert(devmem != -1);

//for analog measurements

dr\_page = (unsigned long)mmap(0, getpagesize(), PROT\_READ|PROT\_WRITE

, MAP\_SHARED, devmem, DATA\_PAGE);

assert(&dr\_page != MAP\_FAILED);

spistart = (unsigned long)mmap(0, getpagesize(), PROT\_READ|PROT\_WRITE,

MAP\_SHARED, devmem, SPI\_PAGE);

assert(&spistart != MAP\_FAILED);

adc\_page = (unsigned long)mmap(0, getpagesize(), PROT\_READ|PROT\_WRITE,

MAP\_SHARED, devmem, ADC\_PAGE);

assert(&adc\_page != MAP\_FAILED);

syscon\_page = (unsigned long)mmap(0, getpagesize(), PROT\_READ|PROT\_WRITE

, MAP\_SHARED, devmem, SYSCON\_PAGE);

assert(&syscon\_page != MAP\_FAILED);

pld\_page = (unsigned long)mmap(0, getpagesize(), PROT\_READ|PROT\_WRITE,

MAP\_SHARED, devmem, PLD\_PAGE);

assert(&pld\_page != MAP\_FAILED);

//iniate analog channels

init\_adc\_channel\_calibration(0);

init\_adc\_channel\_calibration(1);

//intiatlize digital input

fptr=(unsigned long \*)mmap(NULL,getpagesize(),PROT\_READ|PROT\_WRITE,MAP\_SHARED,devmem,0x80840000);

pbddr=fptr+0x14/4;

\*pbddr=0xfC;//d0-1 input rest output

pbdr=fptr+1;

\*pbdr=\*pbdr|0x1C;//set d2 and d3 high as the relay is active low

while(1){

temp\_v=read\_ADC\_channel(0); //temp voltage

write(pipe\_c[1],&temp\_v,sizeof(&temp\_v)); //send voltage over pipe to child

light\_v=read\_ADC\_channel(1);//light voltage

light=getLightLevel(light\_v);//convert light to level 1-5

input=\*pbdr & 0x03; //mask input bits

if((input==1)||(input==0)){//float 1 is high

float1=1;

}

else{//float 1 is low

float1=0;

flag=1; //tank refill will occur, flag for dosing

}

if((input==2)||(input==0)){ //float 2 is high

float2=1;

}

else //float 2 is low

float2=0;

if(float2==1){

\*pbdr=\*pbdr|0x04;//drive d2 high

if(flag){

\*pbdr=\*pbdr&0xF7;//dose

flag=0;//stop dosing

usleep(DOSETIME);//wait with pump on

}

else

\*pbdr=\*pbdr|0x08;//stop dosing

}

else if(float1==0)

\*pbdr=\*pbdr&0xFB;//drive d2 low

if(light<3){

\*pbdr=\*pbdr&0xEF; //turn on relay for light

}

else{

\*pbdr=\*pbdr|0x10;

}

//display float and light levels

printf("float 1: %d, float 2: %d\n",float1,float2);

printf("light: %f, %d\n",light\_v,light);

//wait a while

usleep(2000000);

}

close(devmem);

return 0;

}

//function originally from manufacturer slightly modified for adc calibration

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*DESCRIPTION: Read all five of the EP93xx onboard ADC. Discard the first

\*two samples then save the next 10.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*DESCRIPTION: Figure out the calibration value of the board and store them

\*stored\_cal global variable. In eeprom 2027->2046 are the calibration values. 2027->2036

\*are the zero volt calibration values and 2037->2046 are the 2.5V calibration

\*values. Each calibration value is 16 bits written using little endian

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void init\_adc\_channel\_calibration(){

int i, j, ch, addr;

char buffer[20];

/\* intialize the eeprom \*/

POKE16(pld\_page, (PEEK16(pld\_page) & ~CS\_MSK)); //disable CS

POKE32((spistart + SSPCR1), 0x10); //turn on transmit

while ((PEEK32(spistart + 0xc) & 0x10) == 0x10); // wait for unbusy

while ((PEEK32(spistart + 0xc) & 0x5) != 0x1); // empty FIFO's

POKE16(pld\_page, (PEEK16(pld\_page) | CS\_MSK)); //enable CS

POKE32(spistart, 0xc7); // set SPI mode 3, 8bit

POKE32(spistart + 0x10, 0x2); // divide clk by 2

POKE32(spistart + 0x4, 0x0); // stop transmit

while((ee\_rdsr(dr\_page) & 0x1) == 0x1);//wait for unbusy

ee\_wren(dr\_page);

ee\_wrsr(dr\_page, 0x0); // make eeprom R/W

while((ee\_rdsr(dr\_page) & 0x1) == 0x1);//wait for unbusy

/\* Read in the calibration values \*/

//printf("Calibration Values = \n[ ");

addr = CALIB\_LOC;

for(i = 0; i < 20; i++)

{

ee\_read\_byte(dr\_page, addr, &buffer[i]);

//check if board has stored calibration values

if(buffer[i] != 0xFF)

virgin = FALSE;

addr++;

}

//convert to 16 bit values

j = 0;

ch = 0;

for(i = 0; i < 20; i = i + 2)

{

if(i == 10)

{

ch = 0;

j = 1;

}

stored\_cal[ch][j] = (buffer[i] | (buffer[i+1] << 8));

//printf("(%d, %d) = 0x%x | ", ch, j, stored\_cal[ch][j]);

ch++;

}

//printf(" ]\n");

//make eeprom RO

ee\_wren(dr\_page);

ee\_wrsr(dr\_page, 0x1c);

}

//modified function which will read one analog value at a time based on input modified from manufactuer

//which would measure and print all at once. returns the analog reading.

double read\_ADC\_channel(int ch\_num){

double val, full\_scale;

int cur\_ch;

int result;

/\* intialize the eeprom \*/

POKE16(pld\_page, (PEEK16(pld\_page) & ~CS\_MSK)); //disable CS

POKE32((spistart + SSPCR1), 0x10); //turn on transmit

while ((PEEK32(spistart + 0xc) & 0x10) == 0x10); // wait for unbusy

while ((PEEK32(spistart + 0xc) & 0x5) != 0x1); // empty FIFO's

POKE16(pld\_page, (PEEK16(pld\_page) | CS\_MSK)); //enable CS

POKE32(spistart, 0xc7); // set SPI mode 3, 8bit

POKE32(spistart + 0x10, 0x2); // divide clk by 2

POKE32(spistart + 0x4, 0x0); // stop transmit

while((ee\_rdsr(dr\_page) & 0x1) == 0x1);//wait for unbusy

ee\_wren(dr\_page);

ee\_wrsr(dr\_page, 0x0); // make eeprom R/W

while((ee\_rdsr(dr\_page) & 0x1) == 0x1);//wait for unbusy

//read channel based on input parameter

switch(ch\_num)

{

case 0:

cur\_ch = ADC\_CH0;

break;

case 1:

cur\_ch = ADC\_CH1;

break;

case 2:

cur\_ch = ADC\_CH2;

break;

case 3:

cur\_ch = ADC\_CH3;

break;

case 4:

cur\_ch = ADC\_CH4;

break;

}

//discard first two samples

read\_channel(adc\_page, cur\_ch);

read\_channel(adc\_page, cur\_ch);

usleep(10000);

result = read\_channel(adc\_page, cur\_ch);

//callibration

full\_scale = (((((double)(stored\_cal[ch\_num][1] + 0x10000)

- stored\_cal[ch\_num][0]) / 2.5 ) \* 3.3 ));

if(result < 0x7000)

result = result + 0x10000;

if(virgin == TRUE) //use approximation

{

result = result - 0x9E58;

val = ((double)result \* 3.3) / 0xC350;

}

else //use calibration values

{

result = result - stored\_cal[ch\_num][0];

val = ((double)result \* 3.3) / full\_scale;

}

//make eeprom RO

ee\_wren(dr\_page);

ee\_wrsr(dr\_page, 0x1c);

return val;

}

//function which converts voltage to an understandable level of light

int getLightLevel(double voltage){

if(voltage<=.05)//full daylight

return 5;

else if(voltage<=.25)//overcast day

return 4;

else if(voltage<=1)//very overcast

return 3;

else if(voltage<=1.9)//dark room

return 2;

else //extremely dark

return 1;

}

## Works Cited:

"TS-7250." *ARM-based PC/104 Single Board Computer for Embedded Systems*. Technologic Systems, n.d. Web. 10 May 2015.

<ftp://ftp.embeddedarm.com/ts-arm-sbc/ts-7250-linux/samples/>

"Potential of On-Line Visible and Near Infrared Spectroscopy for Measurement of PH for Deriving Variable Rate Lime Recommendations."*MDPI*. N.p., n.d. Web. 10 May 2015.

<http://www.mdpi.com/1424-8220/13/8/10177/htm>

" CITtinkertecharduinoponics”.sunjournal, n.d. Web. 10 May 2015.

http://www.sunjournal.com/files/imagecache/story\_large/2014/04/26/CITtinkertecharduinoponics042814\_arduino.jpg