Tim Savas
2020-05-15
E123 Final Project Report

Abstract

I am excited to report that I made progress on my project! I have created this final update walking through the process and milestones. Please see my Github and video links along the way. You'll see the assets in my Github as described below.

Assets

The link to the project's Github repository is: **https://github.com/tsavas/e123_project**
Here is a an outline of the assets in the repository for your review:

Asset Folder	Description
pictures	A catalog of photos I took along the way of making the project. I like to produce a visual journal of my projects.
programs	The programs I made for the project. The best ones are humidity_sensing.ino and shoot_move_shoot.ino
reports	A folder of my three reports for the class, including the original abstract, an update midway, and this final report.
spec sheets	A folder of spec sheets for the components I used. The key component was the Telaire Applications Note, thanks to Scott.

Videos

I documented my milestones on video and those moments are linked here. I mention them at the top of each description in this update, too.

Asset Folder	Description
8051_stepper_on	I turn a stepper motor on with a switch on the 8051.
8051_stepper_off	I turn a stepper motor off with a switch on the 8051.
sensor_actuator_demo	Watch me turn a fan on and off based on a sensor value.
shot_sequence_prototype	The first successful test of shoot-move-shoot sequence.
plant_spin	A demo of the turntable spinning a plant quickly.
stepper_close_up	A macro shot of the two turntable gears actuating.
turntable_full_sequence_demo	An in-situ shot of the turntable and camera running.

Milestones

Here are the main milestones I achieved, which I will detail in this project report.

- Wrote C++ code for sensor-actuator loop for the mushroom growth chamber.
- Actuated a stepper motor with the Silabs 8051 and button push, and Arduino.
- Actuated a DSLR camera shutter with the Silabs 8051, and Arduino.
- Wrote C++ code for shoot-move-shoot motion time lapse sequencing.

Mushroom Sensor-Actuator Control Loop

Mushroom Sensor Actuator
video link: sensor_actuator_demo
code file: humidity sensing.ino

Please see the above video for a demo! I was able to get an environmental sensor-actuator control loop going! You'll see in my demo video that I used a temperature and humidity sensor to fire up a small 5V fan. The fan turns on when a temperature threshold is reached and then turns back off when the temperature drops back below the threshold. It's effectively a thermostat.

DHT11 and Fan Temperature Control

I used a DHT11 temperature and humidity sensor for this. I had it as backup in the event that my CO2 sensor was misbehaving or too difficult (more on that in a minute). The DHT11 is a digital sensor, according to its documentation.

In order to use it, I downloaded a corresponding library. Though this was most certainly a shortcut to getting easy readings off of the sensor and onto the Arduino, what's exciting is that I think I know what these "libraries" are actually doing now after studying digital electronics. Data are coming off of such digital sensors in a particular way. Perhaps a single measurement is a byte long, and at the head of that byte is a single bit that itself is a piece of metadata intended to tell its receiving device "this is the start of a new measurement, ready?" I'm speaking in vague terms here, but UART works quite like this, as I have read through the documentation on it. In my mind, UART is almost like a shift register, where you have all of these serial bits coming in from a sensor and onto the microprocessor. But those data are not to be read as single bits—they're probably more like a byte wide (see the specification sheet of the sensor, of course), and again some of those bits are metadata that cue up the microprocessor or sensor, respectively.

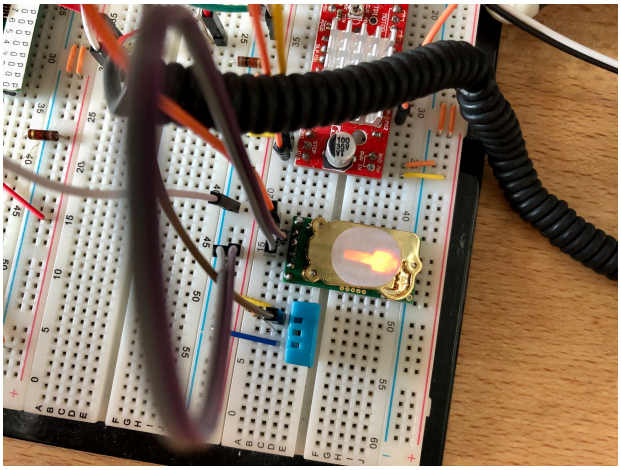
All told, digital sensors require some parsing and manipulating on the part of the microcontroller that is taking in their readings. They're different from analog sensors in this

way. With analog sensors, you can simply read their voltage level, do some conversion specified on the spec sheet, and you're in business.

So my understanding is that these libraries provide the code that is doing this parsing and manipulating. Again I am speaking in vague terms as a result of being economical about how I use my time this weekend (there's a practice exam to take!). I do not mean to advertise a lack of understanding in my attempt to communicate my actual understanding. Just explaining how the concept of "libraries" were effectively demystified for me through our course material, if I have this right.

Temperature Control Code (also in Github file)

```
//Pin connections for Sensor:
// pin 1 : VCC
// pin 2 : DATA
// pin 3 : NC
// PIN 4 : GND

#include <dhtnew.h>                   //include the DHT11 sensor library

int DHT11_PIN = 8;                    //create a variable for the sensor's pin
int FAN_PIN = 7;                      //create a variable for the fan's pin

DHTNEW mySensor(DHT11_PIN);           //create a DHT11 instance and assign it to the pin

void setup()
{
  Serial.begin(115200);               //initialize serial comms and set the baud rate
  mySensor.setHumOffset(10);          //some boiler plate setup for the DHT11 sensor
  mySensor.setTempOffset(-1.5);       //some boiler plate setup for the DHT11 sensor
  pinMode(FAN_PIN, OUTPUT);           //set the fan's pin as an output pin
  digitalWrite(FAN_PIN, HIGH);        //set it high, in off position, at start
}

void loop()
{

    mySensor.read();                          //read the sensor
    int temp = mySensor.getTemperature();     //create a variable and put the temp value in it
    Serial.print(mySensor.getHumidity(), 1);  //print the humidity value with one decimal place
    Serial.print("\t");                       //text break
    Serial.println(temp, 1);                  //print the temperature value

    if(temp > 22){                            //if temp is higher than 22C
        digitalWrite(FAN_PIN, LOW);           //turn on the fan
    }
    else if(temp <= 22){                      //if temp is lower or eq to 22C
        digitalWrite(FAN_PIN, HIGH);          //turn off the fan
    }
    delay(2000);                              //wait two sec until next reading

}

// END OF FILE
```



CO_2 Control
code file: co2_sensing.ino

Alas, I did not get a reading off of my CO_2 sensor in time for this submission. I was trying to do it through UART. I connected Tx and Rx lines between the sensor and Arduino, complementing them of course. Then hooked up the sensor's other pins according its specs. I thought the Tx and Rx pins did the necessary UART parsing magic I needed, since they are the Arduino's designated serial (Rx and Tx) pins, and I assumed that parsing was done with the **Serial.read()** function. But golly, not a byte in site showed up on my serial monitor. Here is my attempt from that angle:

Beginner's CO2 Control Code (also in Github file)

```
int CO2; //create a variable for incoming CO2 data

void setup() {

  Serial.begin(19200);              //initialize serial comms and match baud rate to sensor
  Serial.println("Program Start");  //writint to the console is just fun


}

void loop() {

if (Serial.available() > 0){     //only do this if there's something coming onto the serial
  CO2 = Serial.read();           //move whatever's on the serial over to the CO2 var
  Serial.print(CO2);             //print the CO2 reading
  delay(100);                    //wait one second until the next reading
  }
}
```

CO2 Control After a Better Understanding

9.3 GAS PPM

This command reports the current gas measurement in ppm.

Use the Modbus Read Input Registers function (4) and read one (1) register at address '5003'D ('138B'H).

Modbus Request (UART)

'15'H	Slave Address (default is 21)
'04'H	Function code
'13'H	Starting address (MSB)
'8B'H	Starting Address (LSB)
'00'H	Input registers to read (MSB)
'01'H	Input registers to read (LSB)
xx	CRC (LSB)
xx	CRC (MSB)

Modbus Response (UART)

'15'H	Slave Address (default is 21)
'04'H	Function code
'02'H	Byte count
xx	MSB of the 16-bit data
xx	LSB of the 16-bit data
xx	CRC (LSB)
xx	CRC (MSB)

A very helpful nudge from Scott improved my current working approach to this UART situation. Scott was able to track down a Telaire T6700 Application Note document, which contained far more detail and specifications about the sensor than I was attempting to glean off my initial two-pager. This is where all of the real magic is contained. Indeed like I was saying before about UART [which I wrote last week before receiving this spec document], you have to send a signal to the sensor to get it to send something back to you. And there's a bunch of metadata involved, beyond just the CO_2 measurement, like status of the sensor itself, so you also need to do some parsing of the bits sent back. In short, from here I will continue to work with the sample code from the Application Note to try to get a reading from the sensor—excited about that!

Stepper Motor and Camera Control

Stepper Motor Control using the Silabs 8051
video link: 8051_stepper_off
video link: 8051_stepper_on

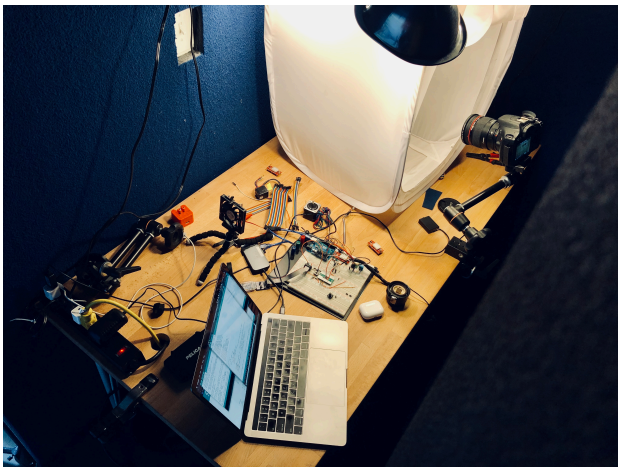
Please see the above video links for a demo! This was the most basic part of my project and yet perhaps one of the most exciting moments. I controlled my stepper motor from the 8051 using a button press, both off and on, as shown in the videos. I used an adaptation of Scott's stepper motor assembly code for this application, changing some libraries and I believe a few other things around. Alas, I did not write comments to my changes when I made them so this Assembly code file is not exactly in perfect form, and again it is an adaptation of my classmate's, but I'm submitting it in any case.

Stepper Motor Control & Camera Control, Arduino Prototype
video link: shoot_sequence_prototype
code file: shoot_move_shoot.ino

Please see the above video link for a demo! I was able to get my shoot-move-shoot operation going with an Arduino! I had many revelations about what I was doing throughout the process. For a refresher on those from my project update from last week:

- What the Enable pin does on a hardware I/O peripheral device and cool ways to use it!
- Why stepper motors always burnt out on me.
- How and why stepper motors can be powered off of their own power supplies.
- C++ code became much easier with an increased understanding of the microcontroller.
- How stepper motor drivers are just edge-triggered devices, which are familiar now!
- How and why to correctly wire up a camera shutter cable.

I cleaned up my code and am submitting the file for this. I switched over to Arduino from the 8051 because I effectively wanted to be able to more easily control variables and parameters of my “shoot-move-shoot” sequence. You’ll see I added those in here, to give me a lot of control over what kind of time lapses I want to shoot. For instance if my objective is to shoot a time lapse of a plant rotating on my turntable over the course of five days, taking 4000 shots total, and turning 900 degrees on the turntable—well I can do that! If I’d like to shoot a time lapse of an ice cube melting, taking 300 shots, turning 270 degrees, over the course of one hour—I can do that, too!



Stepper Motor Control & Camera Control In Situ
video link: turntable_full_sequence_demo
video link: stepper_close_up
video link: plant_spin

Some time back I made a sweet little custom tripod head. It’s a turntable actuated by a stepper motor. But I never really used the darned thing because my electronics chops weren’t quite on part with my mechanical aptitude. Now that I got the shoot-move-shoot sequence operating properly, I connected it to my stepper motor. Oddly enough it was working quite well, with smooth motion with the stepper motor, but just today it started acting funny :/ and I’m currently troubleshooting it. The gears are not getting along with one another, and it seems to have something to do with my driver and delay between pulses. Still it’s pretty neat, check out a few of those videos of it moving at various shoot-move-shoot settings!

I also built a small dark room in my place where I can isolate light. One complication with shooting time lapses of plants growing is “nighttime.” Plants like darkness but cameras, not so much. This darkroom allows me to keep lights on the plants and camera shooting the time lapse 24/7. Woohoo!

Progress Report

Here's a status report on how I moved ahead on the original project scope:

Step	Description
Sensor-Actuator Handshaking	
Spec a CO2 sensor	I found a very nice CO2 sensor off of Digit-Key that will match with my
Spec a fan	I ended up spec'ing a 5V fan just so I could dodge some of the analog complexities that would have been required for higher voltage fans. Easy fix!
Power up the sensor	Did that! It has a really cool symbol that lights up on the top.
Read the sensor	See my notes above on this. I'll get there with the CO2 sensor. For now I'm counting this as a completion since I got the temperature sensor to work.
Move the sensor's value with assembly code	Ended up doing this in Arduino, but I'll count it.
Turn the fan on	Got it!
Turn the fan on with Assembly code, pushbutton	Got this! Since I was using a 5V fan, this was easy to do with the 8051.
Turn the fan on with Assembly code, timer	I skipped this, though it would have been pretty straightforward.
Link the fan and the sensor	Done through Arduino :)
Data logging	
Store a reading internally	I did not get to this part of my project but wow I would love to make this work.
Store a reading at an interval, internally	
Timestamp the data, internally	
Store data onto external memory	
Parse and read the data	
Visualize the data	
A Second Sensor-Actuator Loop	
Repeat these steps for another sensor actuator loop	Soon!
Build an Intervalometer	

Step	Description
Trigger a DSLR camera with the 8051, pushbutton	Yep did that.
Trigger a DSLR camera with the 8051, timer	I did not do this. I jumped over to Arduino before playing with timer's in the 8051.
Move a Stepper Motor	
Spec a stepper motor and driver	Did that. Works great.
Trigger a stepper motor with the 8051, pushbutton	Yes!
Link the motor to a camera rail	Yes! With Arduino.
Create Motion-Based Intervalometer	
Link the motor and the camera	Yes!
Program a sequence for a photo shoot	Modify the time and turn variables in the program according to time lapse I would like to capture. For instance, if I would like to shoot a mushroom growing for one day while turning on a table, write a program that shoots and steps the camera once every three minutes.
Stage the time lapse	Still working on this! This is the fun part :)
Shoot the time lapse	
Post-process	