ECE 445 Senior Design Laboratory

Fall 2017

Individual Progress Report

Automatic Pill Dispenser

Iskandar Aripov (aripov2)

Team Members:
Iskandar Aripov
Mattew Colletti

Instructor:
Prof. Arne Fliflet

TA:
Rebecca Chen

1. Introduction

1.1 Team Project Overview

Our project's goal is to build a configurable pill dispenser unit that interactively collects the input from the user such as specific time and days of the week and dispense selected pill at these specified times. Our pill dispenser can dispense one or two different pills.

1.2 Individual responsibilities and role in the greater project

My role in the project is to write and test all the software for the whole system. It includes interaction between peripherals such as LCD, time unit, buttons, sensors, motors, LED with the microcontroller. Each of the mentioned peripheral needs its individual module to effectively talk to microcontroller and other peripherals such as LCD with buttons and time unit, motors with sensors, etc. I am responsible for any programmable components specific hardware additions such as resistors and potentiometers. I am also responsible for hardware pin configuration for testing and actual design and also for communicating any software specific details that may affect hardware design.

2. Individual design work

2.1 Design Considerations

There are no major design changes since the design review. We decided to use two buttons for navigation process and not to use speaker. For testing we use standard arduino uno board, but for actual design we plan to burn the bootloader onto the actual microcontroller.

2.2 Diagrams

Circuit

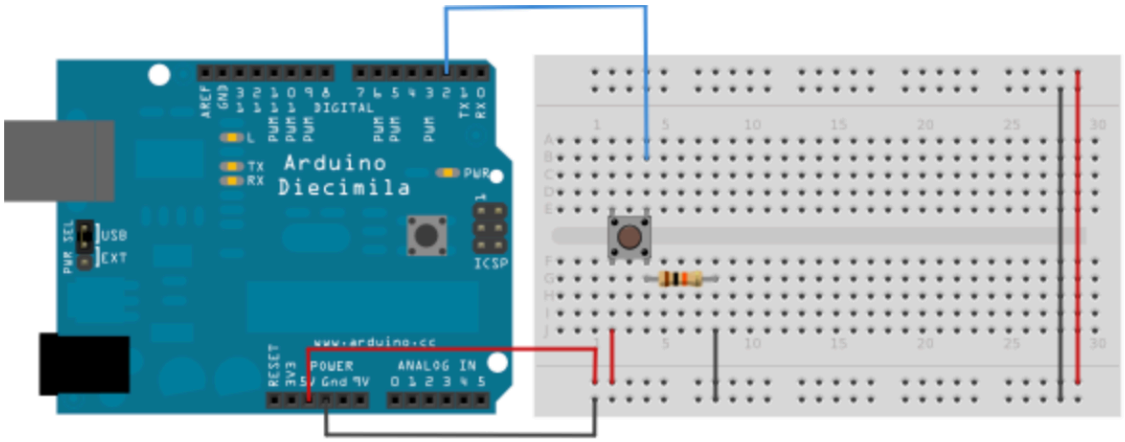

Figure 1: Button Arduino interaction [1]

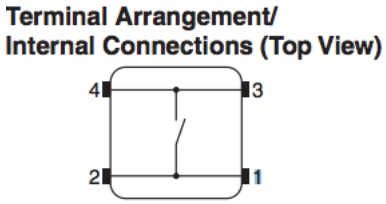

Figure 2: Button architecture [2]

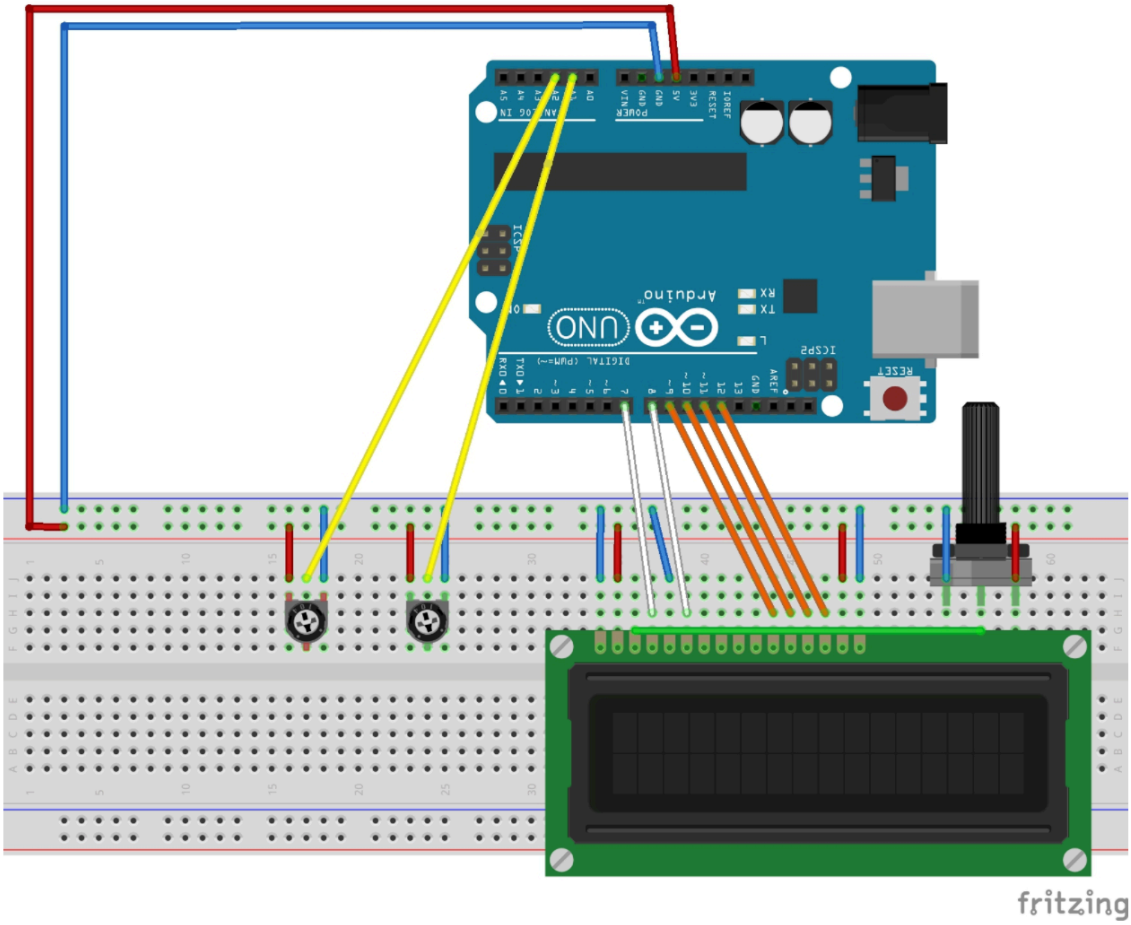

Figure 3: Possible LCD pin configuration. [3]

For LCD we needed to add potentiometer and connect it to pin V0 (3rd pin from left) for characters to background contrast control. We also needed to connect the last two pins (most right ones) to Vcc and GND for brightness.

```
#define STEPS 4096
#define STEPSPERREV 64
#define SENSORPIN 13

time_t t = now();
//printf("50");
//Serial.print(minute());
//Serial.print(second());

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper1(STEPSPERREV, 8, 9, 10, 11);

// the previous reading from the analog input
int sensorState = 0;

void setup() {
  // set the speed of the motor to 30 RPMs
  stepper1.setSpeed(240);
  pinMode(SENSORPIN, INPUT);
 digitalWrite(SENSORPIN, HIGH); // turn on the pullup
  Serial.begin(9600);
}

void loop() {
  // get the sensor value
  sensorState = digitalRead(SENSORPIN);
    if (sensorState == LOW) {
      stepper1.step(STEPS/4);
    }
    else {    // turn LED off:
    stepper1.step(0);
    }
    Serial.print(second());
    delay(1000);
  // move a number of steps equal to the change in the
  // sensor reading
  // remember the previous value of the sensor
}
```

Raw code for Motor Sensor Interaction

Steps = Number of steps in One Revolution * Gear ratio.
Steps = (360°/5.625°)*64 "Gear ratio" = 64 * 64 = 4096.
This value will substitute it on the arduino sketch. Since we want motor to rotate clockwise and counterclockwise we are using range of –Steps/2 to Steps/2. So to invert a motor by 180 degrees we need 180/360 * Steps/2 = Steps/4.

Now the most challenging part of the design is to implement steps described in design document in microcontroller FSM and pseudocode regarding buttons to arduino to LCD interaction and to fetch and store the time and days of the week.
Some obstacles and their solutions include:

- Since buttons can be pressed at any time single digitalRead statement can't capture it without a while loop.
- After the button is pressed at each step we need some sort of tracking system for our cursor and what user tries to do. This is particularly challenging for data collection states such as days of the week and time. For every state we implement a control statements and recursion to original function.
- LCD is slow and microcontroller is very fast relative to the lcd so we need to add appropriate delays in code to allow LCD to process instructions.
- We need to distinguish between the pills as per original pseudocode. This can be accomplished by creating separate global variables for each pill.
- We need to assign the times to the time unit. We are still working on this.

```
void prestep1(){
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Select Pill ");
  lcd.setCursor(0, 1);
  lcd.print("12 ");
 }
int step1(int cursorPosition){
  // Step1

  lcd.setCursor(cursorPosition, 1);

  enterState = digitalRead(buttonEnter);
  rightState = digitalRead(buttonRight);
  while(enterState == LOW && rightState == LOW){
    enterState = digitalRead(buttonEnter);
    rightState = digitalRead(buttonRight);
  }delay(1000);
  if(rightState == HIGH && cursorPosition == 0){
    step1(1);
  }
  else if (rightState == HIGH && cursorPosition == 1){
    step1(0);
    }
   else
   return (cursorPosition +1);
}
```

Raw code for step1, which is "Select Pill" stage on the pseudocode.

```
void prestep2(){
  // Step2
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Select Days ");
  lcd.setCursor(0, 1);
  lcd.print("MTWHFSS ");
}

int* step2(int cursorPosition, int* weekDays){
  lcd.setCursor(cursorPosition, 1);
  enterState = digitalRead(buttonEnter);
  rightState = digitalRead(buttonRight);
  while(enterState == LOW && rightState == LOW){
    enterState = digitalRead(buttonEnter);
    rightState = digitalRead(buttonRight);
  }delay(1000);
  if(rightState == HIGH && cursorPosition < 7){
    step2(cursorPosition+1,weekDays);
  }
  else if(rightState == HIGH && cursorPosition >= 7){
    step2(0,weekDays);
  }
  else if(enterState == HIGH && cursorPosition >= 7){
    return (weekDays);
  }
  else if(enterState == HIGH && cursorPosition < 7){
    if (weekDays[cursorPosition]==0){
       lcd.setCursor(cursorPosition, 2);
       lcd.print("Y");
       lcd.setCursor(cursorPosition, 1);
       weekDays[cursorPosition]=1;
       step2(cursorPosition,weekDays);
    }
    else if(weekDays[cursorPosition]==1){
       lcd.setCursor(cursorPosition, 2);
       lcd.print(" ");
       lcd.setCursor(cursorPosition, 1);
       weekDays[cursorPosition]=0;
       step2(cursorPosition,weekDays);
    }}}
```

Raw code for step2, which is “Select Days of the week” stage on the pseudocode.

(Note: other states raw code won't be posted because "enter time" step is much larger in size and "verify entry" and "another pill?" are similar to step 1.)

2.3 Testing/Verification

2.3.1 Test plans and procedures with specific timelines

November 7 2017	Test states communication and information fetching of LCD, microcontroller, buttons by giving input with buttons and checking result of microcontroller operation on lcd.
November 7-November13 2017	Test time unit integration to the whole system by making sure that time unit accurately counts time,and notify microcontroller when it is equal to setup time, add LED to the system and check whether it is on during appropriate time. Debug and software-hardware communication problems
November 13-November 20 2017	Bootload code into microcontroller, Test the whole system on microcontroller

2.3.2 Modular testing or System testing

Each of the steps on the pseudocode from design document is separately tested to make sure it outputs a correct result. For example for step 1, which is "Select Pill" state, the right button press needs to switch the cursor to the right and record its new pill number by adjusting the position. For step 2, which is "Select Days of the week" when we press enter it shows that that day was selected and if we press enter again it gets deselected. If we press right button it navigates to the next day until we navigate after Sunday entry and press enter. We test each code unit by giving our custom input and comparing the output of the system with what we expect. Each of the steps gets individually tested and after all modules behave as expected we integrate all the modules.

We applied the same procedure for the motor sensor interaction. When sensor gets obstructed, the motor is tested to rotate amount we set in code in particular direction. When we test the time unit individually for functionality and make sure it operates correctly we would be ready to integrate all our code parts into one code. After our code is working under arduino, we will combine all parts with other hardware and our actual microcontroller. Such modular testing can simplify our design debugging.

2.3.3 Results

As of November 6, 2017 all steps perform as expected and output correct results to LCD. The only issue is when user selects another pill option and then selects the same pill as before. When we tried it created overlap between previous timing parameters. The motor is also configured correctly and interacts with sensors in the right way.

2.3.4 Contingency plans for failed tests

To fix aforementioned error we need to clear the data if user selects the same pill after “another pill” state but not alter it if user selects another pill. We would have to come up with the appropriate control mechanism to erase data depending on current and next state pill selection and use recursion after “another pill” state.

3. Conclusion

3.1 Self-assesment

3.1.1 Individual workload vs total workload for the project

I am responsible for the great deal of the project. The software part is rigorous, consists of many modules and is appropriate in difficulty for the rigor of Senior Design. My lab partner is responsible of the hardware part, of which design details I am not fully sure. Since I am Computer Engineering major and he is Electrical Power Engineering, the responsibilities reflect individual major.

3.1.2 Ahead or behind of schedule

We forgot to add time unit to the schedule on design document. Since it had to be replaced and arrived late we would have to deal with it this week. Otherwise, we are on schedule.

3.2 Plans for remaining work

Timeline is similar to design document except adding of time unit to the design of our system. Our ambitious goal is to have a working version by Thanksgiving break. Realistically, we may need to use Thanksgiving break for further any debugging and suggested modifications.

4. Citations

[1] “Button.” *Arduino Tutorials - Button*, Arduino, 28 July 2015, www.arduino.cc/en/Tutorial/Button.

[2] Omron, “Tactile Switch,” B3F datasheet, July 2010. http://omronfs.omron.com/en_US/ecb/products/pdf/en-b3f.pdf

[3] van den Berg, Wimpie. “Displaying Sensor Values on LCD.” Arduino Project Hub, Arduino, 14 Nov. 2016. http://create.arduino.cc/projecthub/Guitarman1/displaying-sensor-values-on-lcd-c0c44f