Crazy Alarm Clock

LAKSHMI MEYYAPPAN

JAMES KAYE

WILLIAM DIEHL

CONG CHEN

Overview

Problem: Some people hit snooze excessively every morning rather than getting out of bed

Solution: Creative snooze tactics

- Tactics will escalate with each snooze

Features

Military clock

Set alarm time

Set minutes between snoozes

Choose tone

Turn alarm on/off

Snooze

Stages

Play repeating tone

Win a simple game of Simon

Run away

Project Division

Main programmer – Will

- Building a state machine for the stages and features of the program
- Setting up the real time clock and alarm/snooze set features
- Combining all parts of the project into one program

Secondary Programmer – Coco

- Setting up interrupts and polling
- Debouncing buttons and switches
- Interfacing with the piezo and setting up the tone set feature

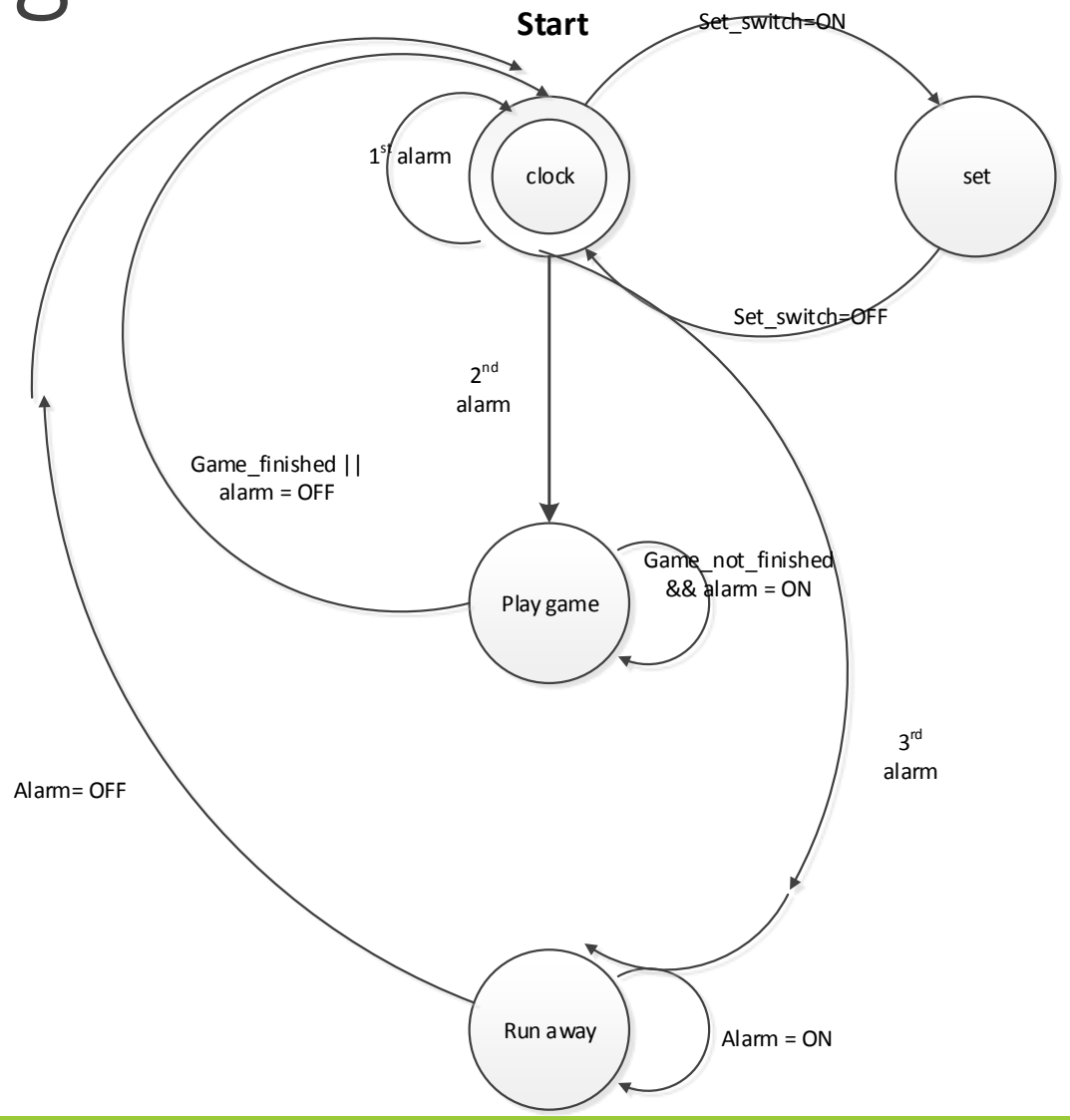
Robotics Lead – Lakshmi

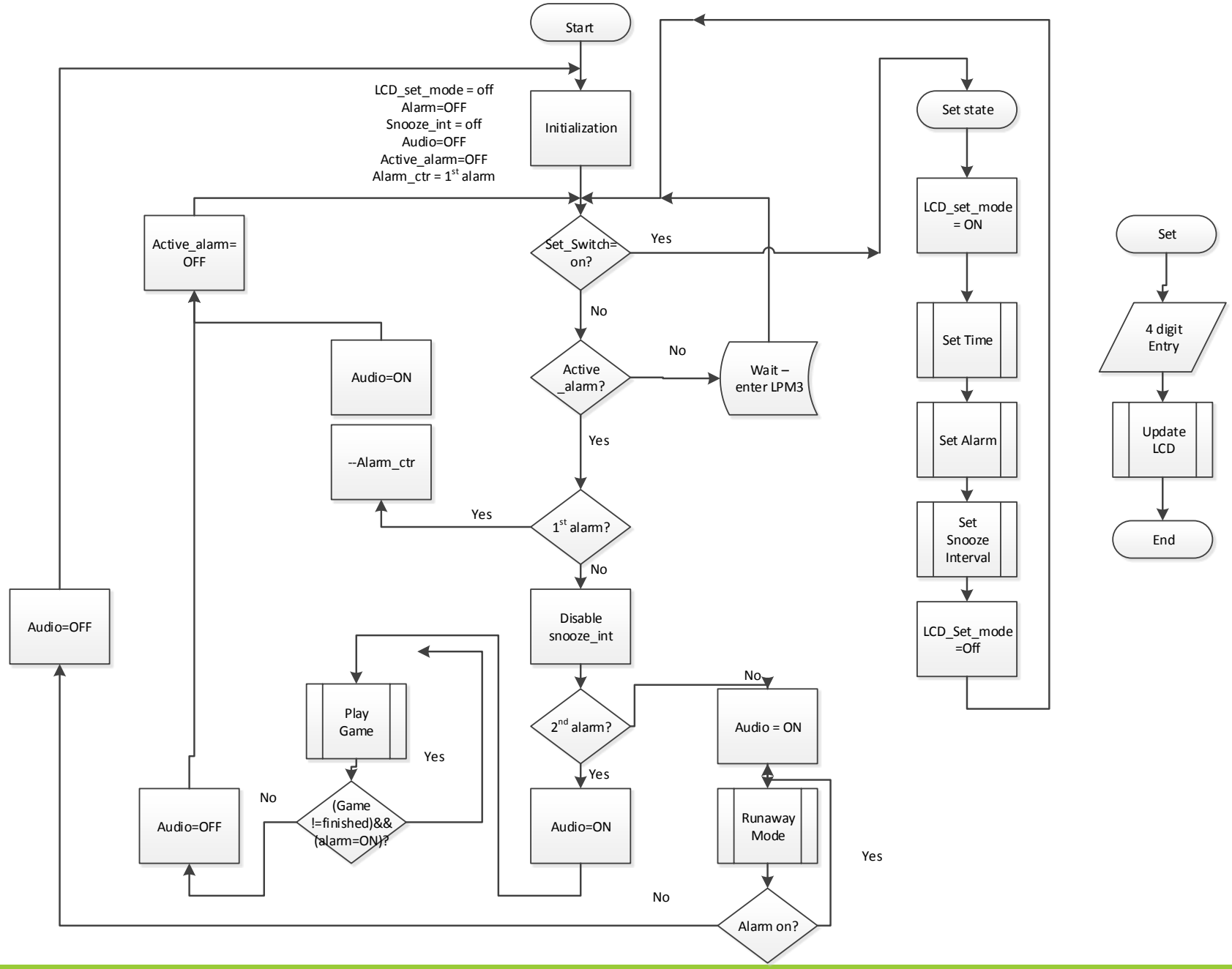
- Driving motors with H-bridge
- Connecting sensors and creating rules
- Building enclosure and attaching wheels

User Interface Lead – Alex

- Interfacing with the keypad/Trellis
- Interfacing with the LCD display and creating views for each stage
- Building the stage two puzzle/game

Main Program Flow




Start
LCD_set_mode = off
Alarm=OFF
Snooze_int = off
Audio=OFF
Active_alarm=OFF
Alarm_ctr = 1st alarm
Initialization
Active_alarm= OFF
Set_Switch= on?
Yes
No
Audio=ON
Active _alarm?
No
Wait – enter LPM3
--Alarm_ctr
Yes
1st alarm?
Yes
No
Audio=OFF
Disable snooze_int
Play Game
Yes
No
Audio=OFF
(Game !=finished)&& (alarm=ON)?
2nd alarm?
No
Yes
Audio=ON
Audio = ON
Runaway Mode
Yes
No
Alarm on?
Set state
LCD_set_mode = ON
Set Time
Set Alarm
Set Snooze Interval
LCD_Set_mode =Off
Set
4 digit Entry
Update LCD
End

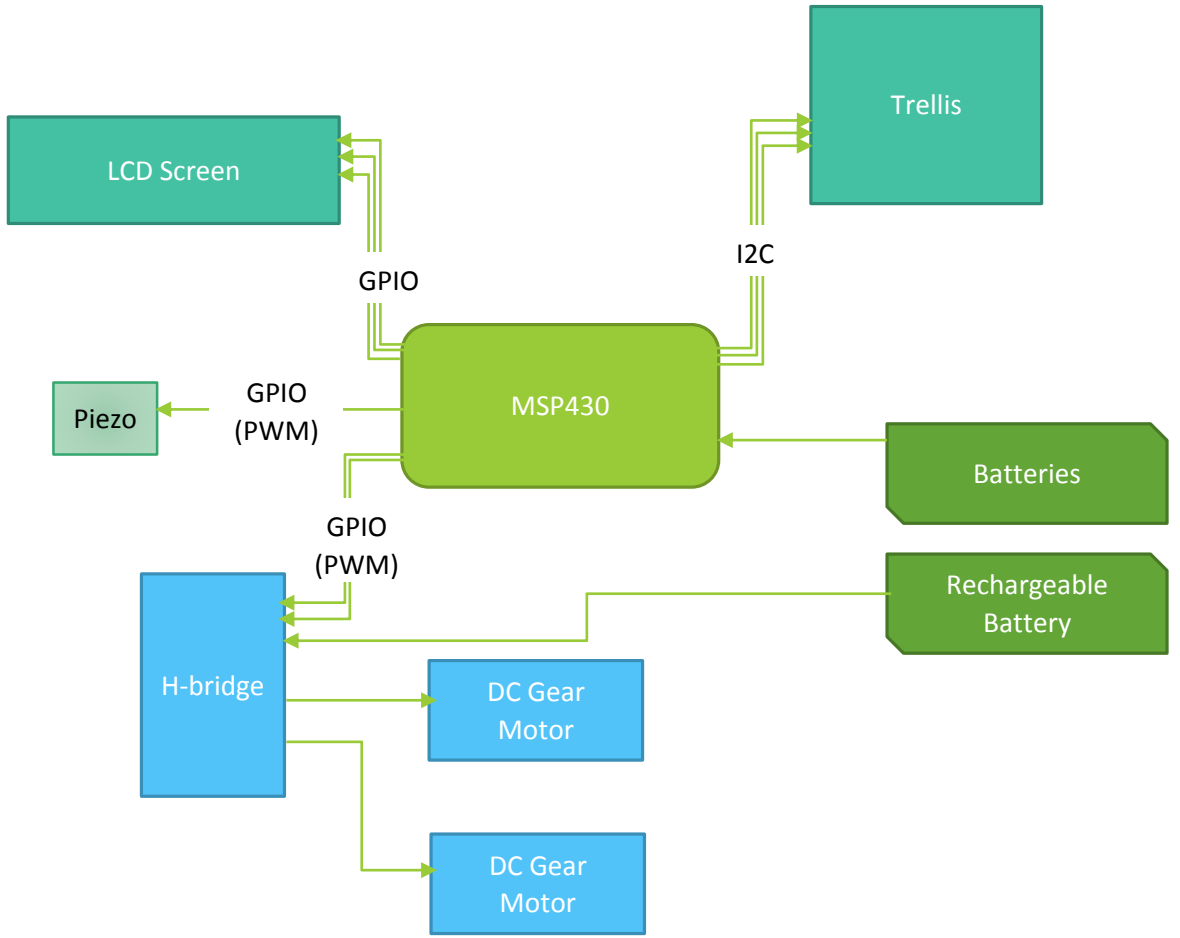
Use of Resources

Real Time Clock (RTC_A) set to ACLK
- Allows waiting in LPM3

Timer_A set to ACLK
- Used for PWM for piezo buzzer

Timer_B set to ACLK
- Generates pulses for tones on Trellis Key Pad
- Generates PWM for H-Bridge

I2C used for Trellis Key Pad



32-Character LCD Screen

Interface:

- 4-bit or 8-bit bus, plus 3 control lines
- Writing a single 8-bit character to the screen involves writing two successive 4-bit nibbles to the bus. Screen then automatically increments to the address of the next character

Progress:

- Breadboarded with all necessary power and control lines
- No code written yet

Challenges:

- None
- MSP430 Sample code for this display is available online!

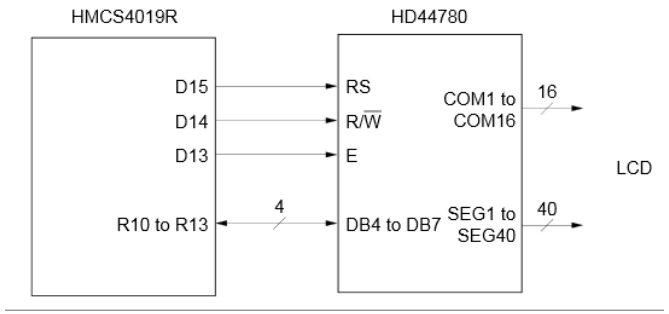
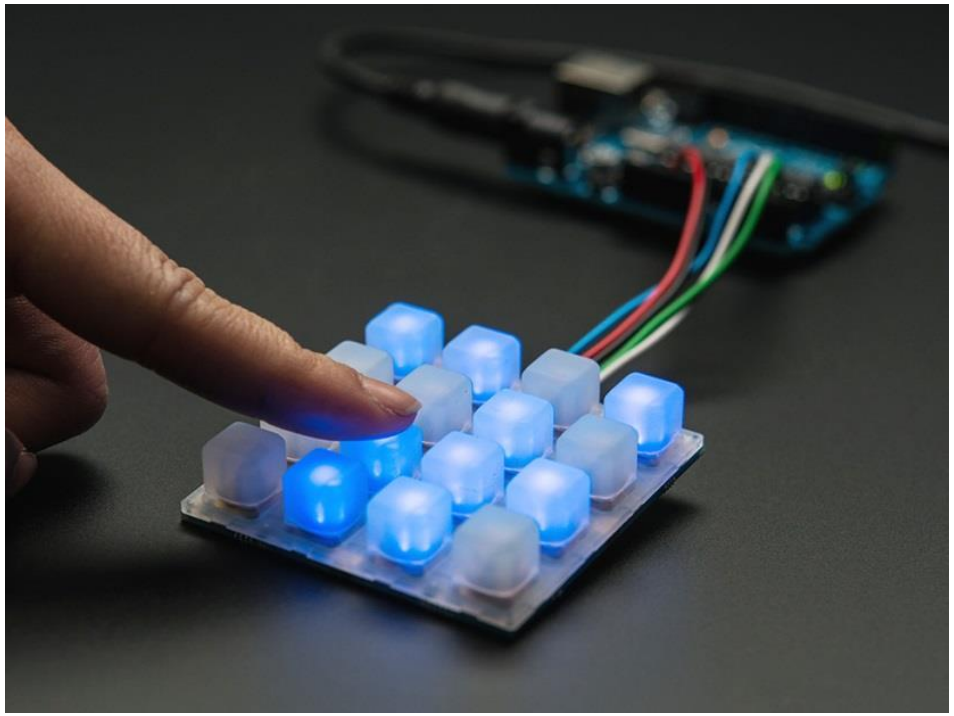


Figure 18 Example of Interface to HMCS4019R

Adafruit Trellis 4x4 LED Keypad



Interface:

- I2C Bus, with optional hardware interrupt line to detect button presses
- I2C address is set by physical jumpers on the PCB
- MSP430 will write to the I2C bus to light up LEDS; read from the bus to detect button presses

Progress:

- Board and buttons are ordered and received. Still need LEDs.
- Code not yet written

Challenges:

- It turns out 3mm LEDs are harder to find!
- No readily available sample code, but we can reverse-engineer the available Arduino libraries

Piezo Buzzer

Interface

- One pin to square wave, the other pin to ground
- Voltage from 3V to 30V

Progress

- Start or stop a square wave to piezo
- Quicken or slow down its frequency

Challenge

- Frequency works at first, but does not work once the start or stop function is applied
- We need to generate a sequence of frequencies that match up to a musical scale

DC Gear Motor

Interface

- Driven by a DC voltage between 3 and 12V
- DC voltage can be drive in either direction to change direction of motor

Progress

- Motors acquired and tested on variable voltages in both directions
- Driven by batteries so it is mobile

Challenges

- Mostly hardware challenges, such as connecting them securely to wheels
- Finding the offset between the two motors so the device drives straight

Dual H-Bridge

Interface:

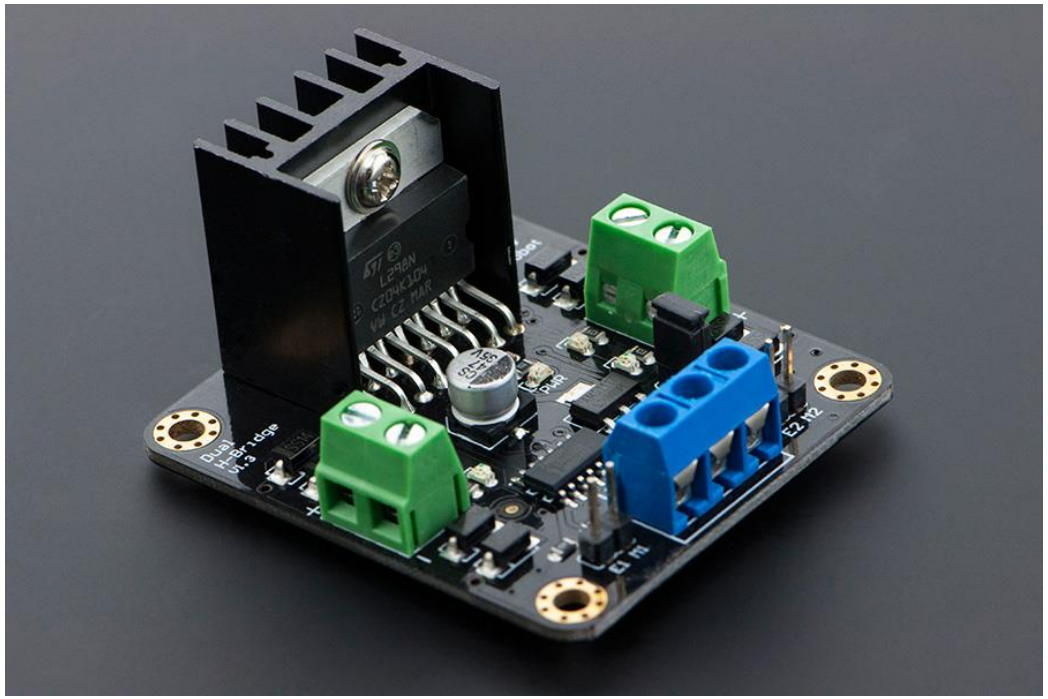
- 3-12V enable and direction pins to GPIO ports
- PWM square wave to motor control pins to change speed
- Motor power source directly from battery

Progress:

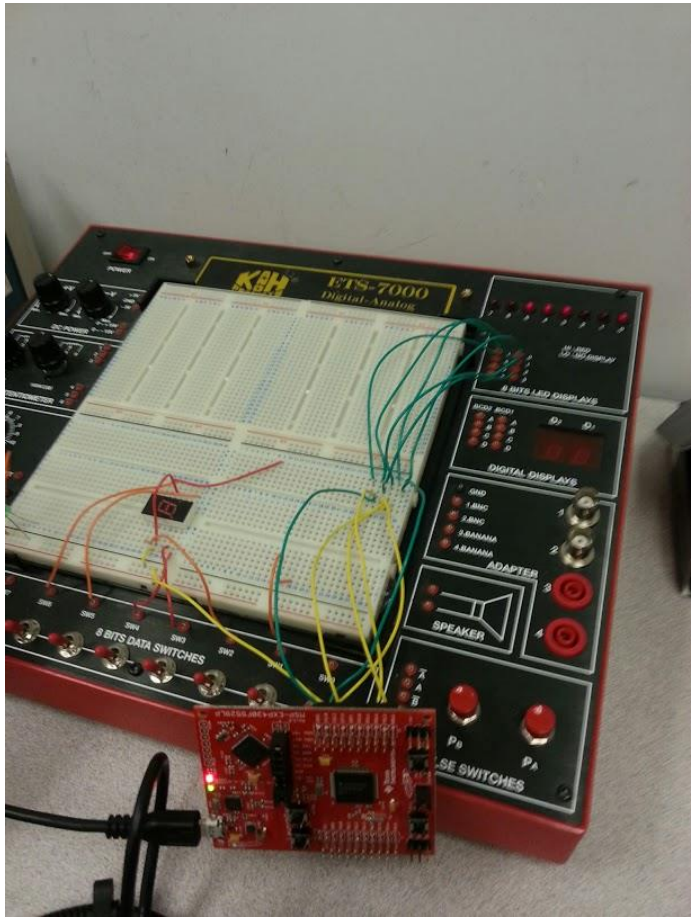
- Driver is talking to MSP430, but only enable is currently connected
- PWM code not yet written

Challenges:

- Very little documentation, so basic control was a lot of trial and error
- Motor offset will have to be taken into account as a percentage of every speed



MSP 430 Progress



Learning the MSP430 to verify that timers work, interrupts occur, and GPIO pins function correctly.

For example, we use the RTC_A for an interrupt-driven binary clock displayed on LEDs (until we get the LCD display routine to work)

Overall Progress

All parts have been acquired

Most parts have been interfaced with the MSP430, some have been tested for desired functionality

Next steps

- Program each part with its own working function
- Build chassis and enclosure

Backup Plans

Each of our components has its own plan B

If we can’t build our own enclosure, we will use an existing chassis to test

If the device cannot roll off a table safely, we will build a ramp or keep it very close to the ground