

James Breen

Han Yang

ATTITUDE +

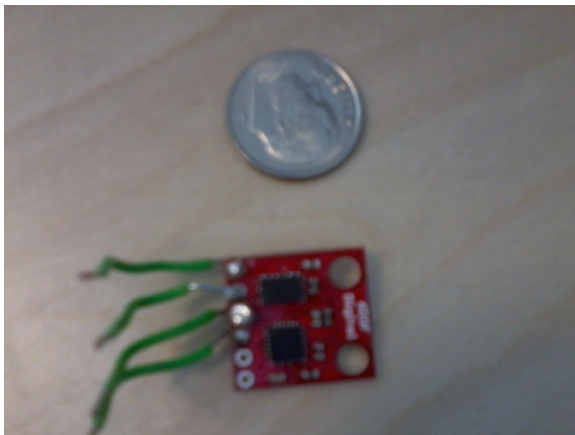
Objective

- Develop a quad copter that is able to stabilize itself on a ball joint located at its center
- The quad copter was attached to a ball joint restricted the movement to pure rotation and simplified the control problem.
- Its called Attitude + because the quad copter only controls it's attitude and it is shaped like a “+”

Inputs and outputs to PIC

Input

- A 3-axis gyroscope was the only sensor used.
- Angular rates were retrieved from gyro with I2C communication protocol.



Outputs

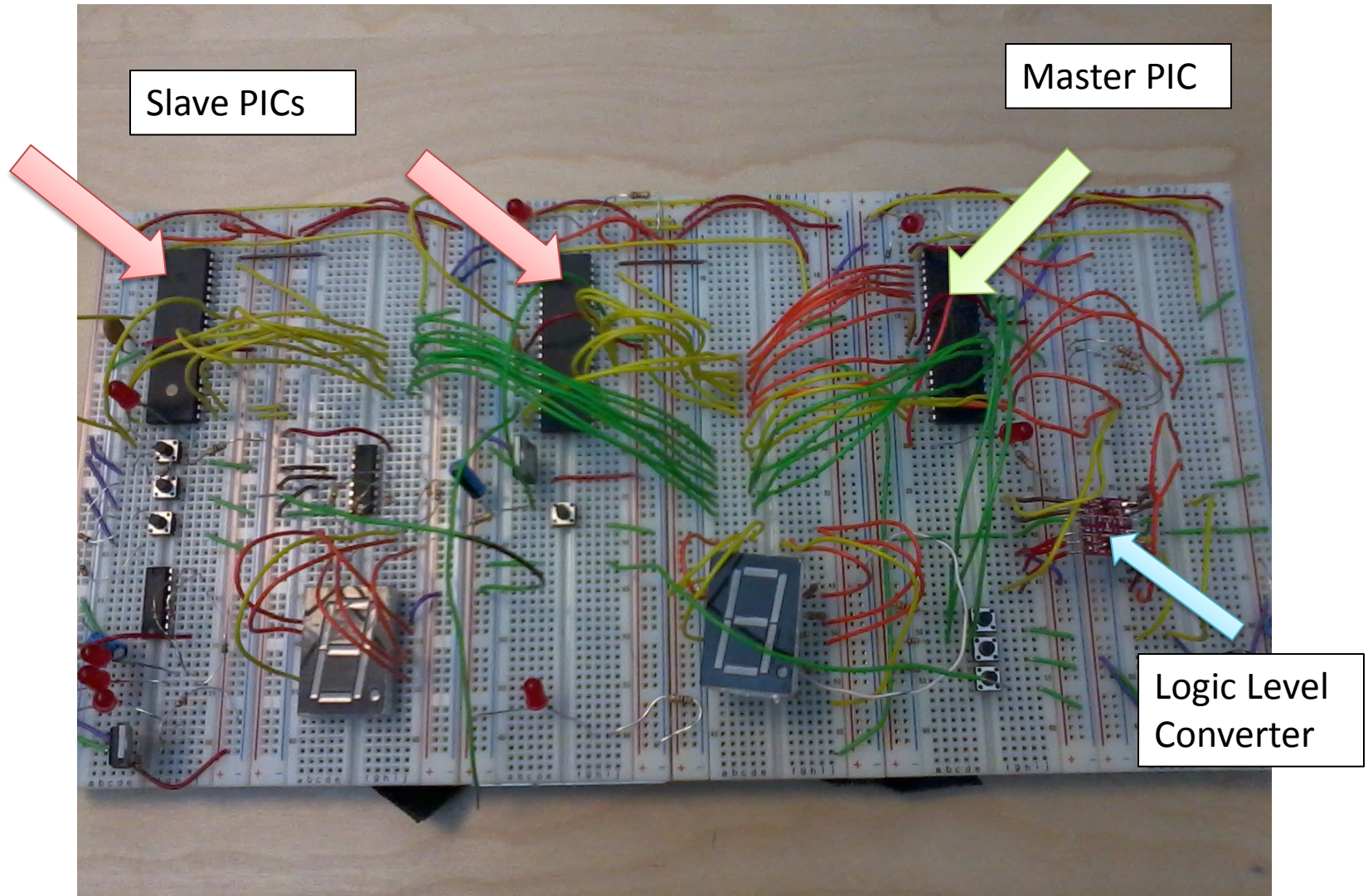
- Four brushless DC motors were used to stabilize the quad copter.
- Thrust output of motors was determined by gyro readings



PIC setup

- 3 PIC18F4520s were used
- 1 Master PIC
 - Retrieved data from gyroscope
 - Performed control calculations
 - Sent data to Matlab (if desired)
 - Commanded Slave PICs to give motors thrust
- 2 Slave PICs
 - Each has 2 PWM channels and controlled 2 motors
 - Each responded to commands from the Master PIC

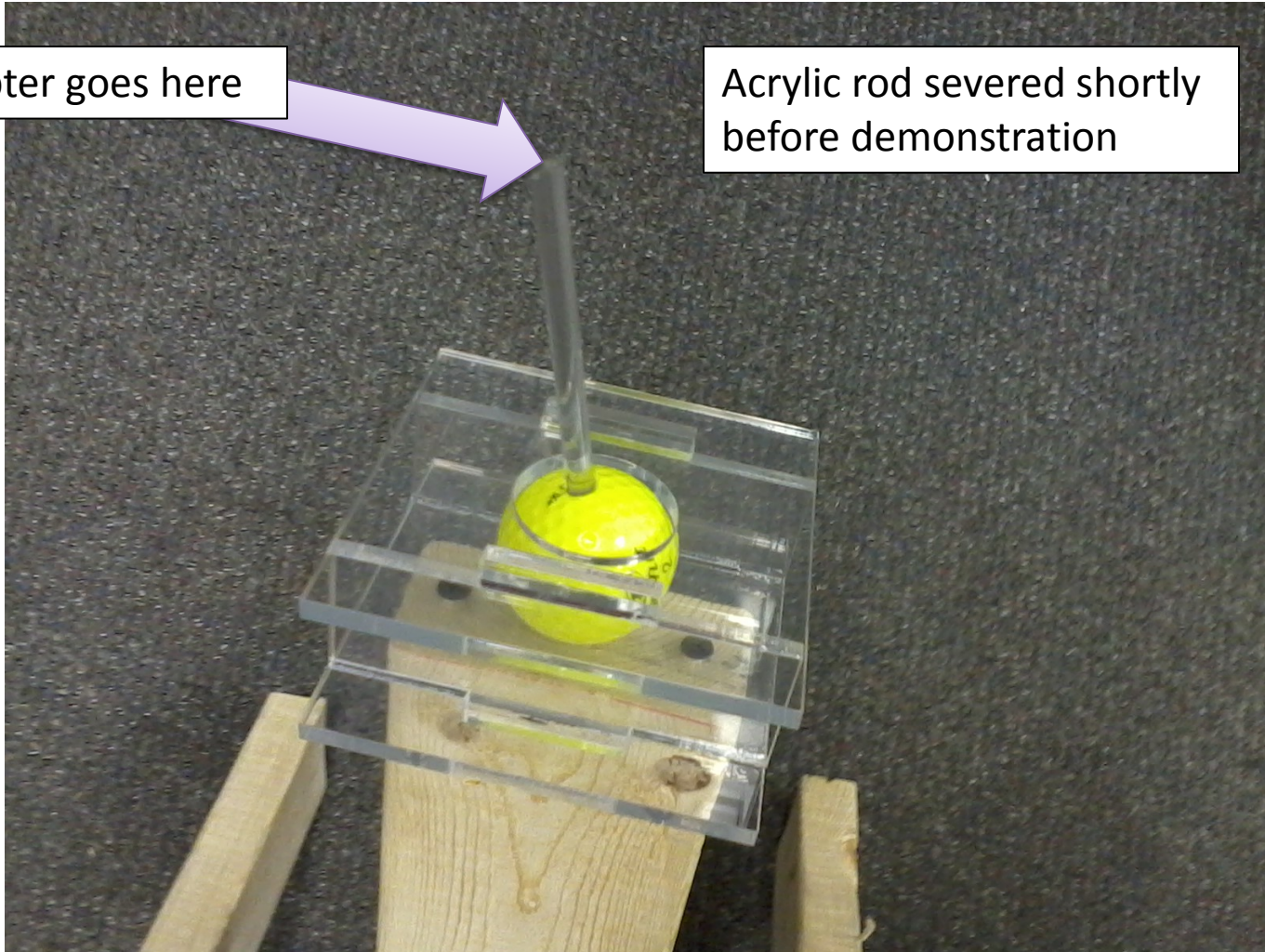
PICs



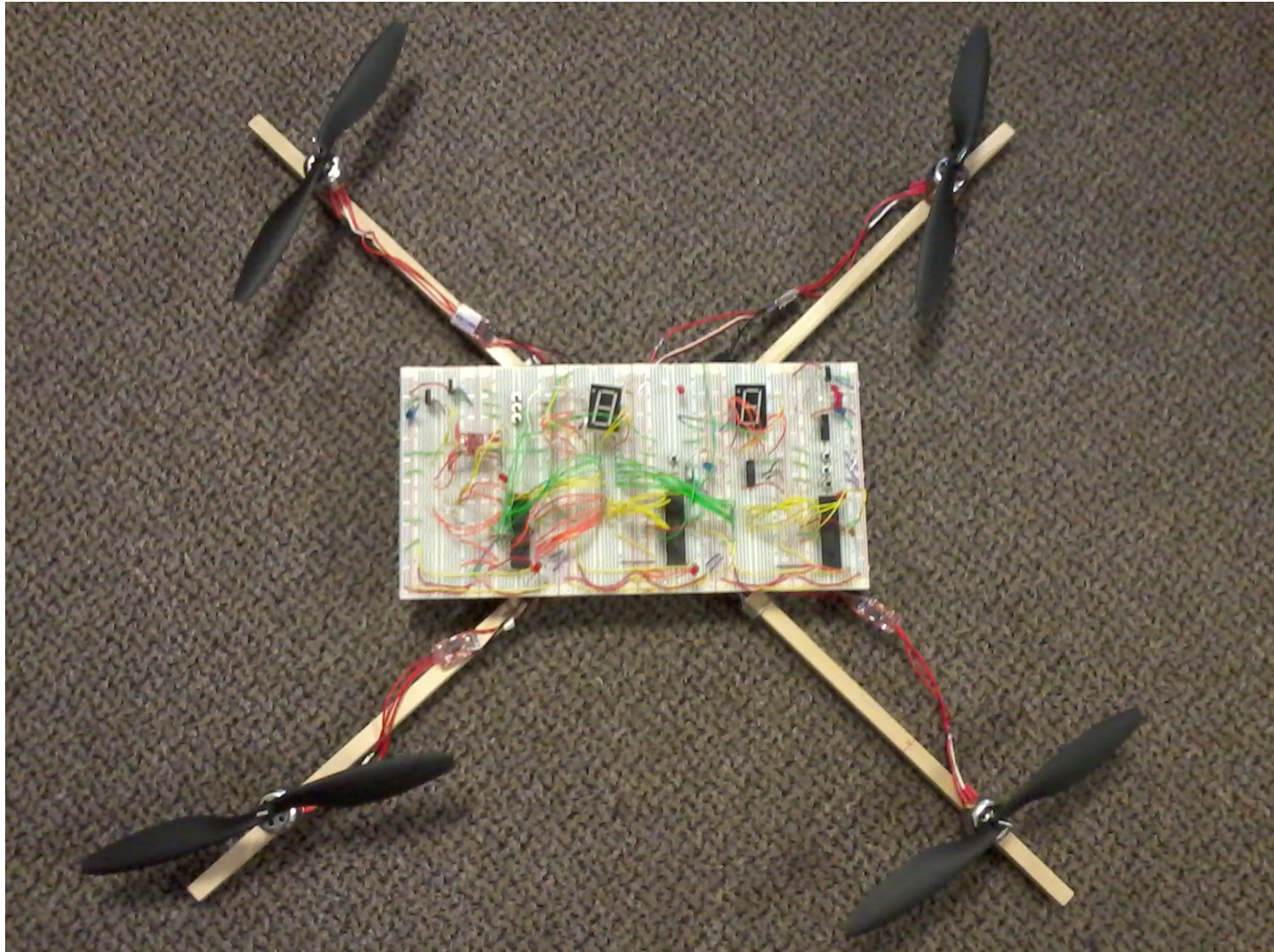
Pivot Stand

Quad copter goes here

Acrylic rod severed shortly
before demonstration



Assembled Quad Copter



Results

Successes

- Data retrieved from gyroscope and plotted in Matlab
- Able to command each of the 4 motors to output any thrust in the motor's range

Failure

- Unable to use gyroscope data to stabilize the quadcopter

Advice

- Concentrate on getting something working and focus on fancy stuff later
- Make sure the mechanical design is sturdy and materials chosen are rigid enough for the application
- Start Early
- MPLAB C-18 Library has some terrific I2C functions
 - EEByteWrite
 - EERandomRead
- 3 great sites to for I2C slave and master code (source code with explanations):
 - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en011798
 - http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en023445
 - <http://www.mikroe.com/forum/viewtopic.php?t=8106&highlight=i2c+slave>
- A few good forums for Brushless motor/Quad copter info
 - <http://www.rcgroups.com/forums/showthread.php?t=1006721>
 - <http://www.rcgroups.com/forums/showthread.php?t=768115>

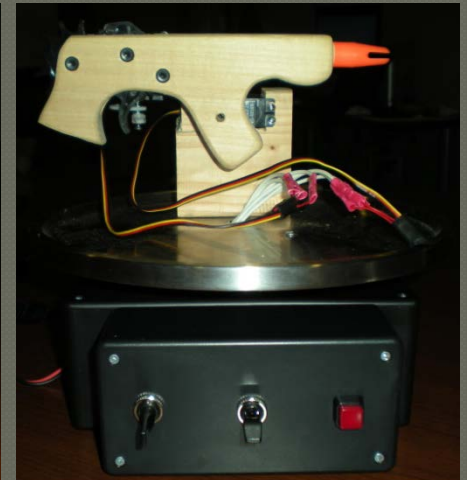
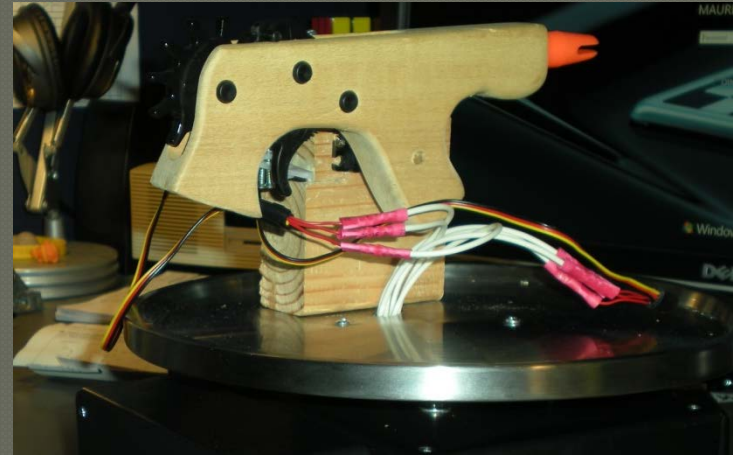
Automated Rubber Band Gun Shooter



By:
Eric Myers and Jordan Maurer

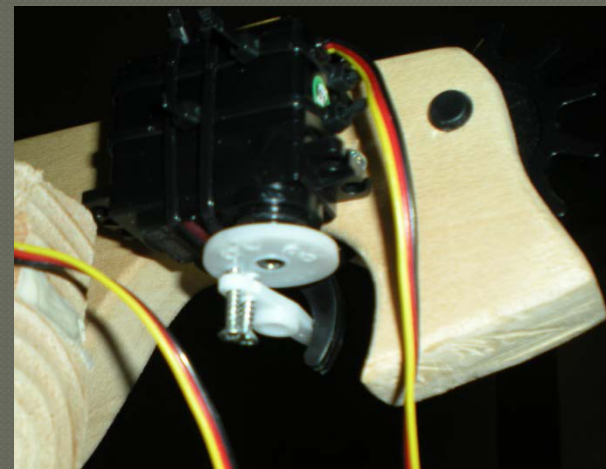
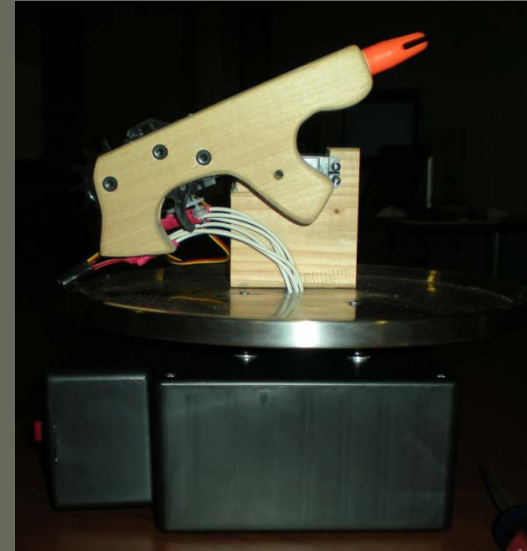
Project Objectives

- Mount rubber band gun on top of a table
- Rotate table to move gun horizontally using a switch



Project Objectives (cont.)

- Angle gun up and down on top of table using a switch
- Fire gun by pushing a pushbutton



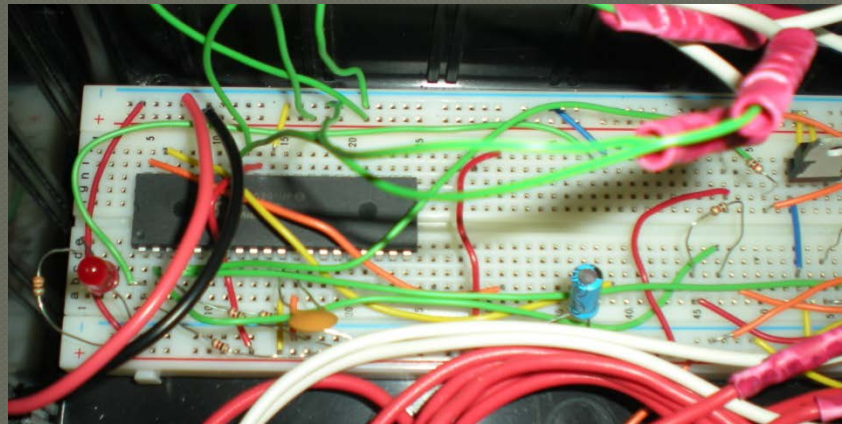
Inputs

- Move horizontally – sensed by a two way momentary switch
- Angle vertically – sensed by a two way momentary switch
- Fire gun – activated by a push button



PIC Use

- Timers – we had two timers overall, one to run each of our servo motors. The timers would overflow at opposite times of each other, but at the same frequency to correctly run the servos.
- Interrupts – The timer overflows would cause an interrupt that would cause an impulse in the voltage for the corresponding servo to act as a pulse width input to the servo.



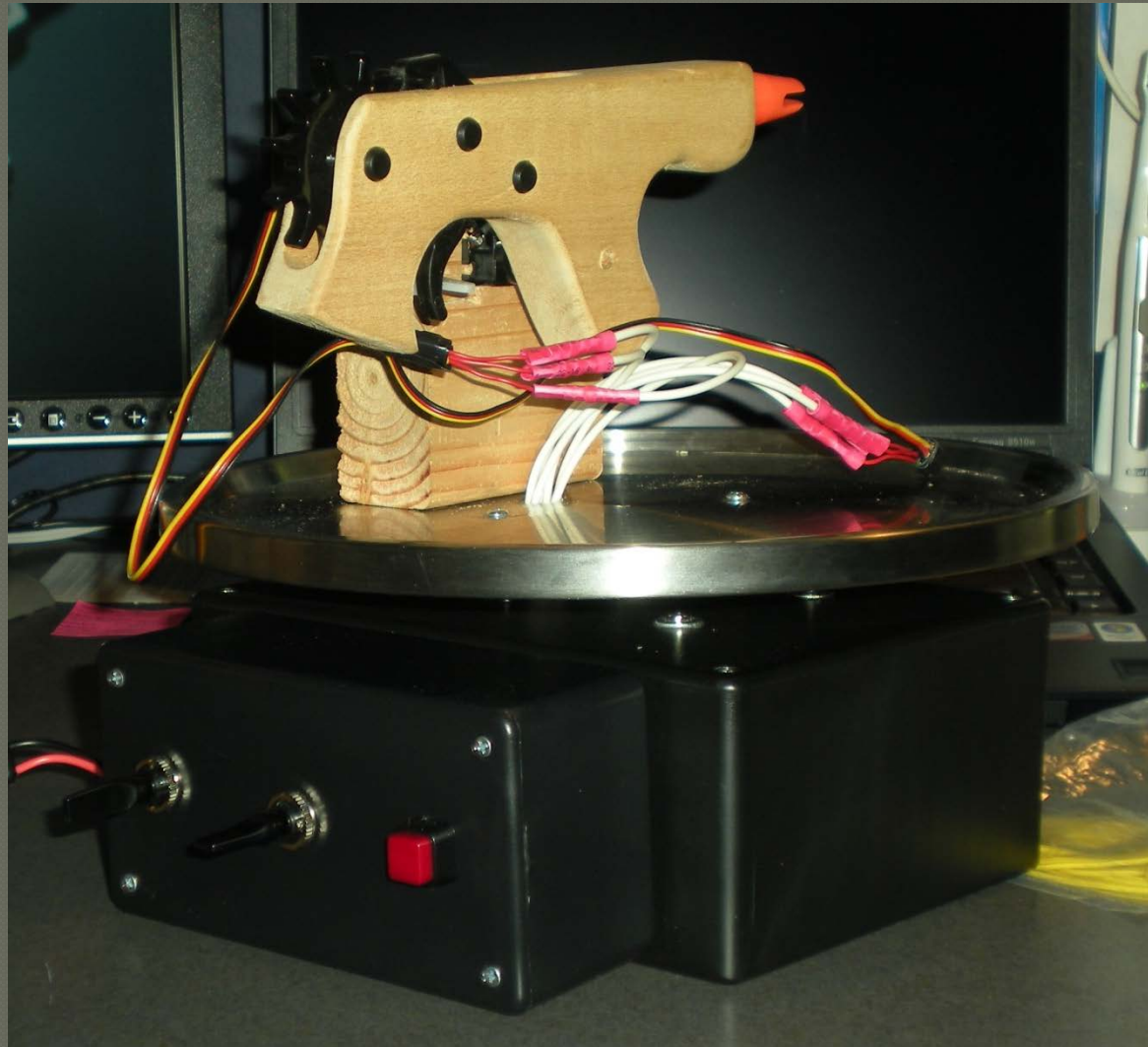
Outputs

- Move gun horizontally – wired a two way momentary switch, so that the motor would run either way based off of a two way momentary switch and power supply alone.
- Angle gun vertically – used a variable called delay to set the pulse width going to the servo. Pressing up on the corresponding switch would increase the delay and angle the gun up, and vice versa for pressing down.
- Fire gun – programmed a sequence that would turn a servo motor to one position that pushes the gun trigger, then goes back to its original position releasing the trigger

Results

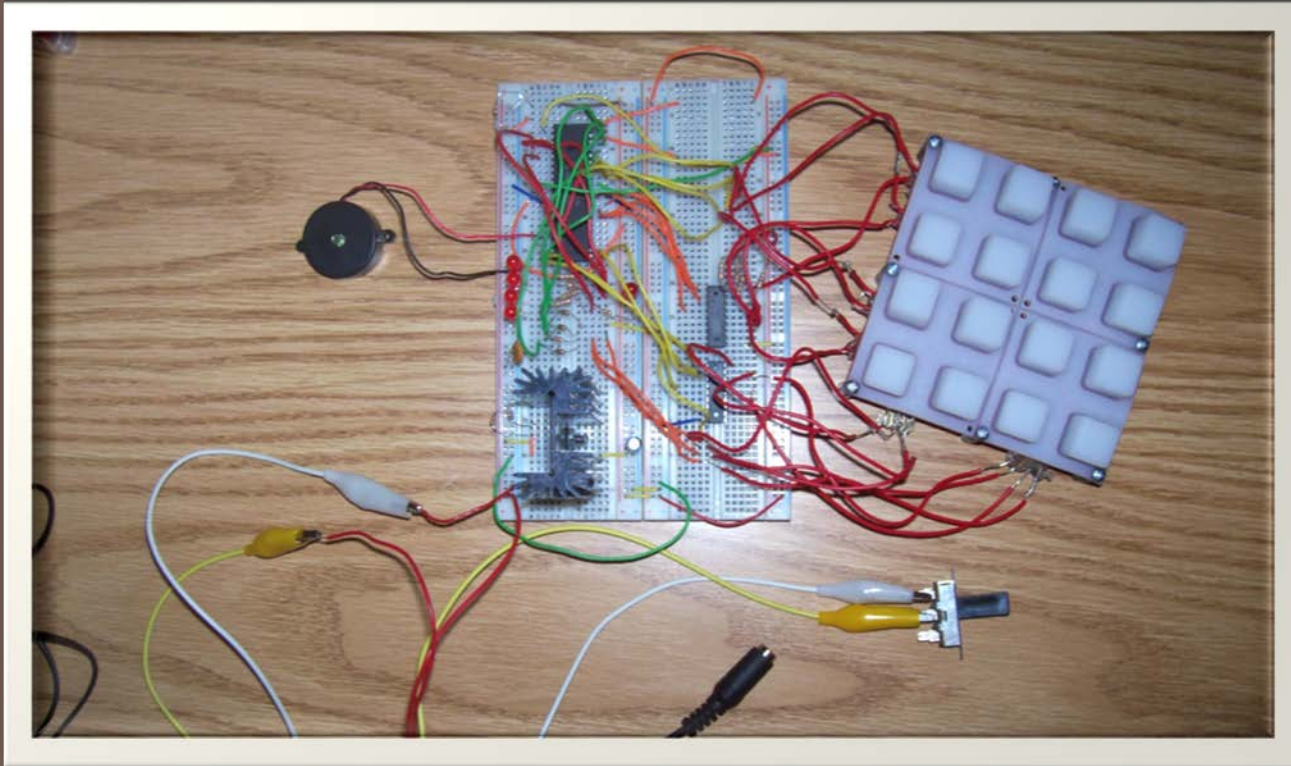
- Our device worked as exactly as planned.
- The rubber band gun angled up and down in increments based on input from a two way switch.
- The rubber band gun turned side to side based on input from a two way switch.
- The rubber band gun fired a single rubber band due to a sequence activation from pressing a one way switch.

Final Product



Future Advice

- Making a solid connection between wires through solder or wire clamps will help the pic program work properly throughout the build process.
- Label wires on both ends when connecting everything so that there is no confusion about which wire goes where and no electrical item accidentally gets fried.
- Test parts early, so you can order new parts if necessary due to lack of working.



MECHATRONICS SYNTHESIZER

Jason Block and Michael Zero

The project objective was to build a fully functioning music synthesizer.



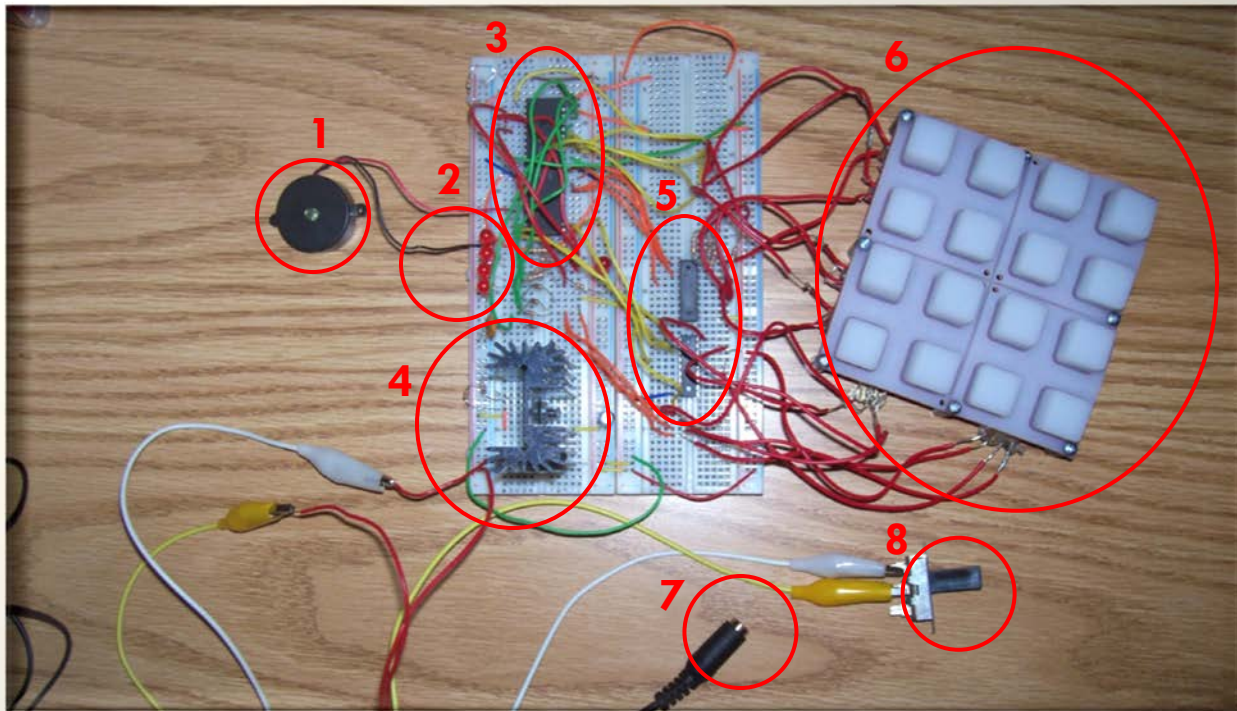
The goal of the synthesizer was to have a button pad that the user could play with to produce a unique tone based on selected buttons.

Pushing a button would turn an LED on underneath the button which showed that the button was engaged. Pushing the same button again would turn the LED off and disengage the button.

The synthesizer would cycle through checking each column of buttons. Based on which buttons were engaged, a tone would emit from a Piezobuzzer one column at a time.

The synthesizer would continuously cycle through each column changing the buzzer tone based on changes made to the button array.

Some parts used were LED's, Darlington's, a button pad, and wire.



1. Piezobuzzer
2. LED's
3. PIC Chip
4. Voltage Regulator and Heat Sink
5. Darlington Chips
6. Button Pad, Buttons, and LED's
7. 12v Power Source
8. Power Switch

Aside from the numbered parts, two breadboards, various resistors/capacitors, and a lot of wire were used. The wire underneath the button pad required cutting, soldering, and hot gluing.

The input from the user was the combination of buttons pressed.

The buttons were laid out as a 4x4 grid. The buttons in each column were associated with one another.

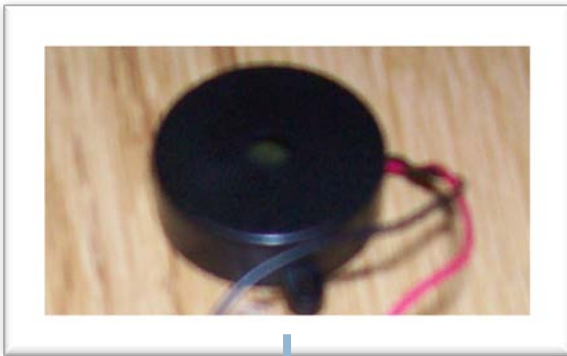
Combinations of pressed buttons in columns caused varying outputs.



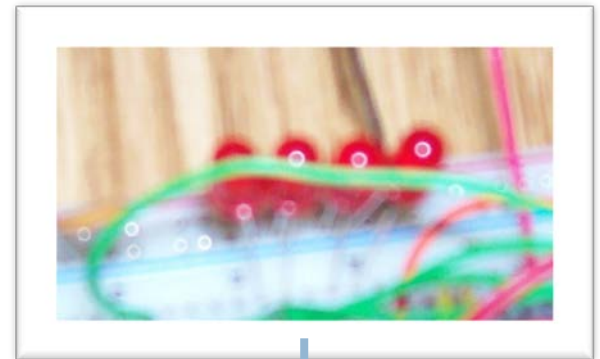
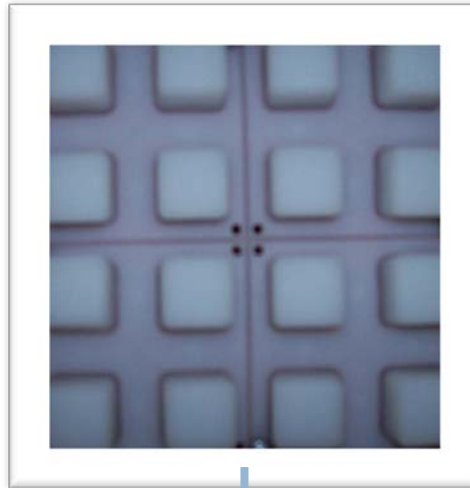
The rows and columns were connected to Darlington chips which were wired to PIC input pins.

The user could pick any of the 16 buttons to push at any time.

The outputs of the system included lights and unique music tones.



The sound was produced from a Piezobuzzer. The sound was unique to the selected buttons.



When a button was pressed on the pad, an LED was turned on. As well, one of four LED's on the circuit board indicated which column of the board was being checked.

The PIC reads information from a button pad grid to process inputs.

In a continuous while loop, each column of the button pad is being checked. Each column and row of the button pad is assigned a certain PIC pin. When the user turns a button on, it goes high. As the synthesizer checks a column, several if statements are in place. If that column and a row are both high, the PWM will be set to play a tone for a designated time until the next column is checked. If multiple rows are high in a certain column, a different tone will be set.

The trickiest aspect of the project was writing the code. It took a multi-step process to accomplish the task.



```
graph LR; A[Develop Overall Code Structure] --> B[Implement Ideas in Code Outline in MPLab]; B --> C[Edit Ideas once in MPLab]; C --> D[Troubleshoot Code with Hardware]; D --> E[Refine Code Continuously];
```

Develop
Overall Code
Structure

Implement
Ideas in Code
Outline in
MPLab

Edit Ideas once
in MPLab

Troubleshoot
Code with
Hardware

Refine Code
Continuously

The project worked mostly but not all features functioned properly.



The synthesizer cycled through columns of the button pad. As it cycled, it checked the button selections and output a unique tone.

Reason for Success: The code evolved from an outline to a solid structure to working tool.

Lesson Learned: Develop a coding strategy, implement, and refine.



The button pad hardware was very touchy. In some instances if you pushed one button, an adjacent button would also be turned on.

Reason for Failure: The team determined it was a hardware issue but did not know how to fix it.

Lesson Learned: Leave ample time to tinker with purchased hardware.

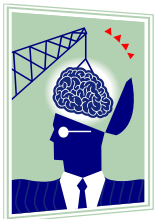
Future ME430 students need to consider time, aesthetics, and scope.



The time it requires to complete the project is easily underestimated. As well, the time required to listen to lectures outside of class and complete lab work can pile up at the end of the quarter. The students need to complete work on the project gradually throughout the entire quarter.



If the students have enough time to make the project look aesthetically pleasing, this needs to be part of the initial planning. It is easy to forget that certain wires that will be soldered need to be long enough to accommodate a nice-looking box. Include the hardware's final resting place in the planning stages of the final product.



The scope of most projects is probably too ambitious at the beginning of the quarter. It is important to really sit down and think about what can be completed in ten weeks on top of lectures, labs, and other coursework. Sometimes the simplest projects end up being the best because they can be completed fully and well.

MOTOR DRIVEN STRING INSTRUMENT

Dan Brooks

Andrew Horvath

Project Objective

Build a simple string percussion instrument using motors controlled by push buttons

Motors should be adjustable to control the force with which the hammers are swung

String are tunable to allow different notes to be played

Inputs

- ▣ Activation of Motors – Initiated by push buttons
- ▣ Duty cycle of PWM – Controlled by potentiometer
- ▣ Motor On-time – Controlled by potentiometer

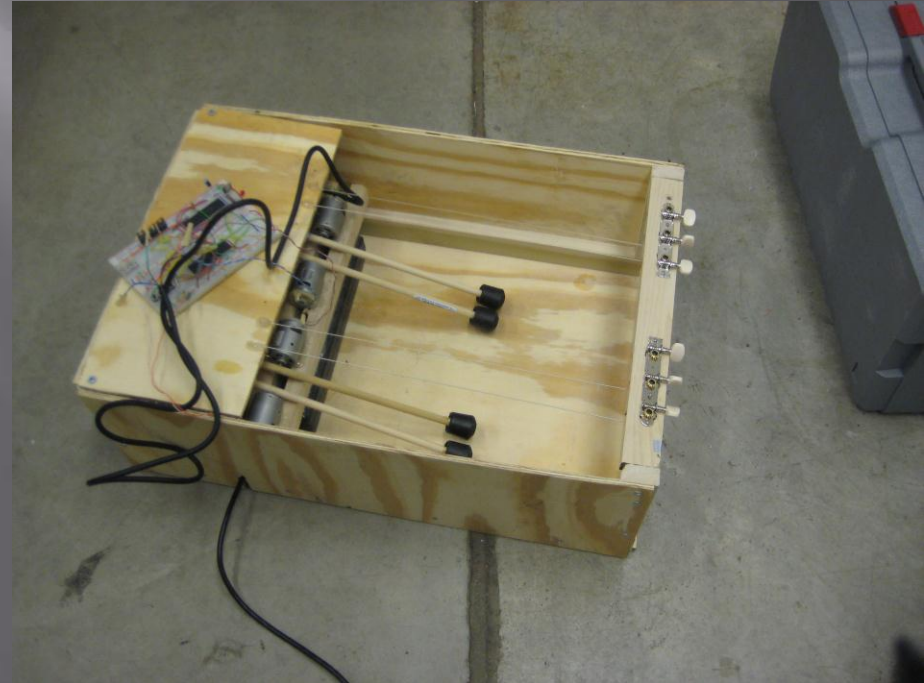
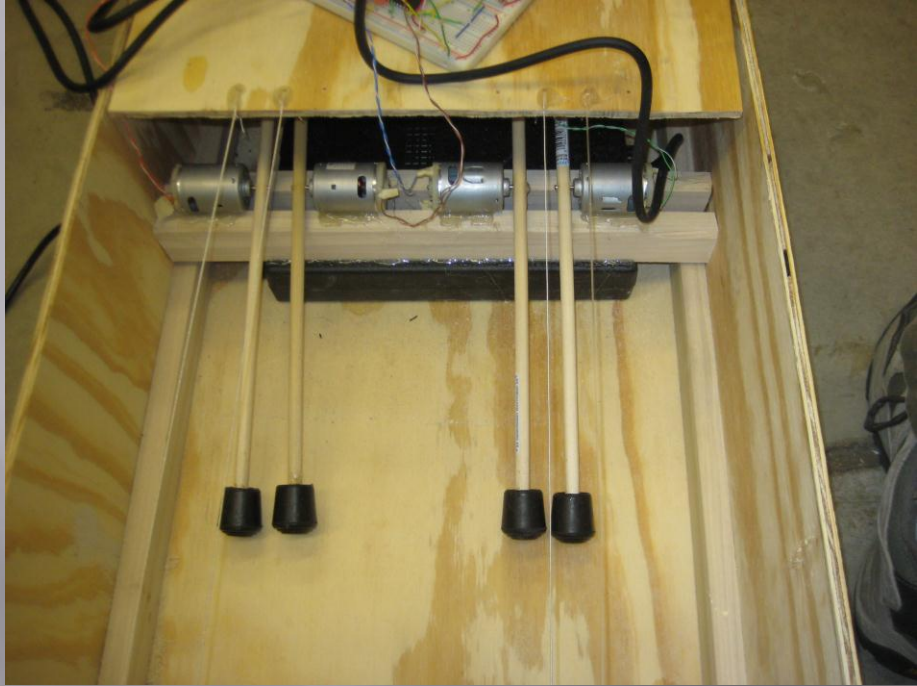
PIC Control

- ▣ Initiation of Motors controlled by interrupts on the button presses
- ▣ Motor speed controlled by PWM duty cycle using ADC with a potentiometer as the input
- ▣ Motor on-time controlled within a predetermined range using ADC with a potentiometer as the input

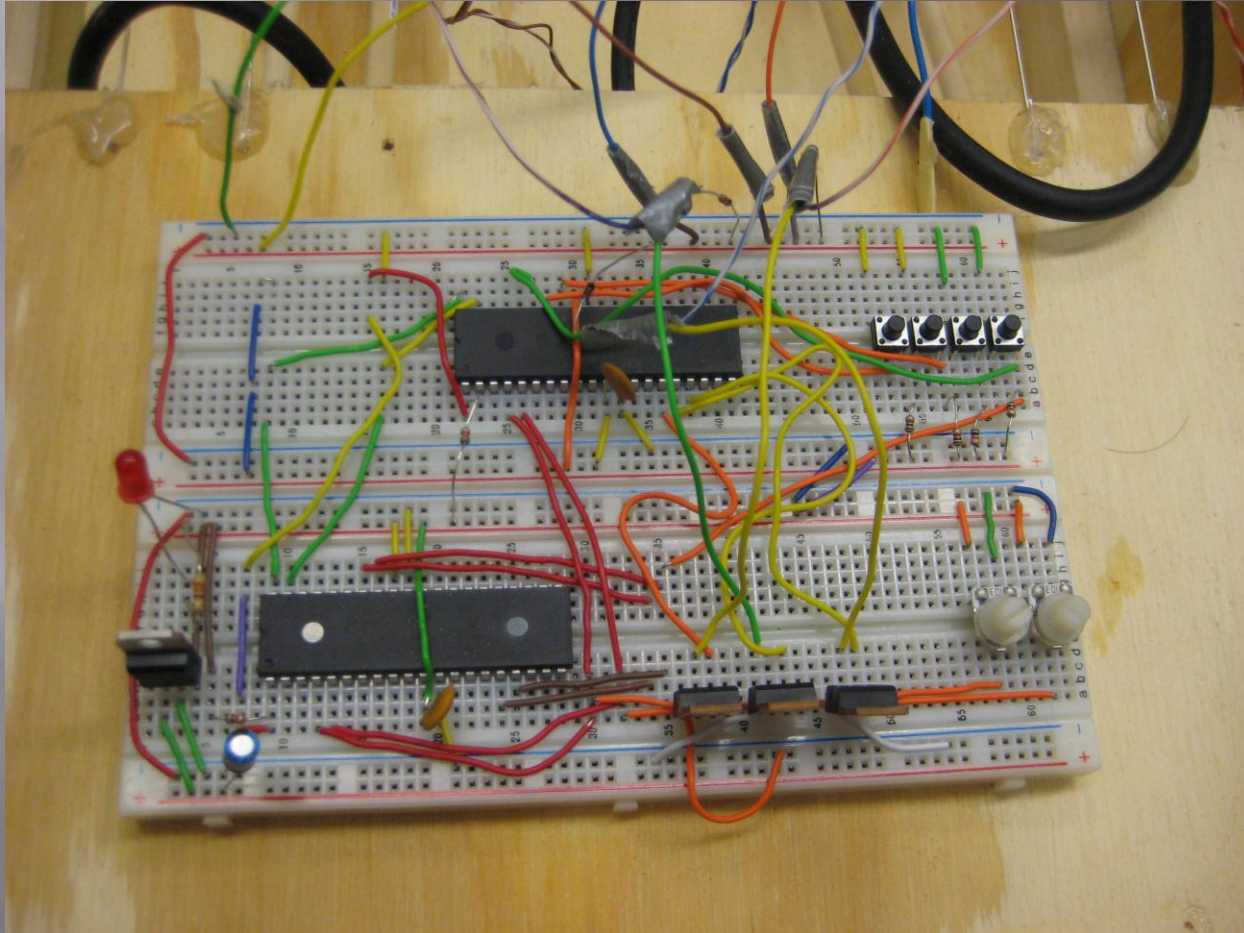
Outputs

- ▣ Motor Activation

Hardware



Circuit



Results

- ▣ The motors moved just as we planned with the exception of one that burned out a MOSFET
- ▣ The PWM was determined to be unnecessary because max power was needed

Lessons for Future Students

- ▣ Make sure to get heat sinks for MOSFETS if you plan on running 24VDC motors with reasonable torque

Mobile Weather Station

ME430 – Mechatronics

George Carpenter

Luke Woolley

Project Objectives

Construct a weather station that:

Accurately measures:

- Temperature
- Humidity
- Barometric Pressure
- Wind Speed

Can be easily transported

- Small
- Portable
- Durable

Requires minimal know-how

- Easy to read
- Easy to use
- Easy to store

Inputs



On/off Switch

- Toggles unit on/off to save battery



Push Button

- Cycles through measurement modes



Humidity/Temperature Sensor

- Accurately measures humidity & temperature



Pressure Sensor

- Accurately measures pressure

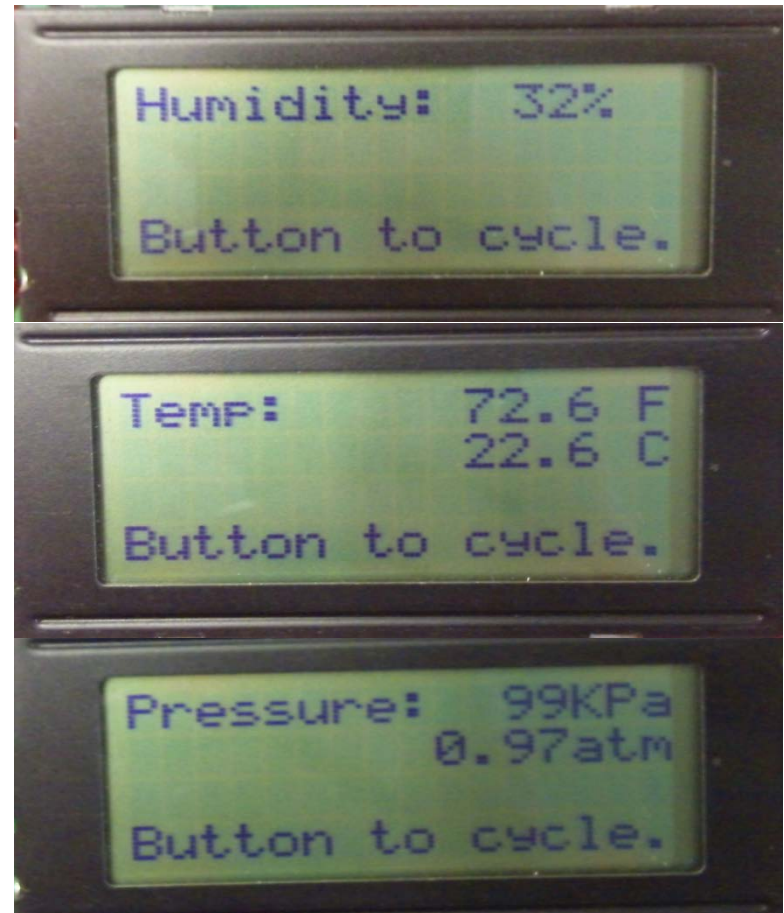


Wind Generator Kit

- Uses motor to generate voltage from wind

Outputs

- LCD Display
 - Shows measurement depending on mode
 - Button press cycles through modes



PIC Use

- ADC:
 - Analog signal from potentiometer, pressure sensor & wind generator
- Interrupts:
 - Used to change mode upon button press
- LCD Module
 - Used to display information on the LCD
- Delays
 - Used to periodically refresh measurement on display

Results



*iPod shown purely for size comparison, not required for operation of weather station

Results

- Fully-functional
 - Accurately measures:
 - Pressure
 - Temperature
 - Humidity
 - Can measure wind speed
 - Requires further calibration to provide accurate results
 - Portable
 - All-in-one unit
 - Battery-powered
 - Durable
 - Made of hard plastic
- Potential Upgrades
 - Build battery compartment
 - No need to open up unit and expose breadboard to change batteries
 - May be able to contain wind generator components for better portability
 - Add better connectors to wind generator
 - Make set-up easier
 - No need for alligator clips
 - Build an anemometer
 - Motor is limited to max 5V output due to PIC

Advice

- Order spares if possible (you never know what will burn out)
- Have an idea how components work before ordering them
- Test code in pieces to make sure each component works
- Modularize your program
 - Helps isolate problems
- Use scraps/freebies whenever possible
- Your best resource is your professor!
 - But you can also cruise the internet for people who have created similar projects

The Party Box

Emma Carroll
Stephan Lemmer

ME430 – 01
Olson
Winter 1011



Project Objective

- ◎ This device is the perfect solution for individuals who want to have the ability to host an instant party.
- ◎ Our Objectives
 - > Have a spotlight and disco ball respond to CDS sensor and an override switch
 - > Have x-mas lights respond to microphone and an override switch
 - > Ambient light detection control with potentiometer
 - > Ambient sound detection control with potentiometer
 - > Main power controlled by a switch

Party Box Components

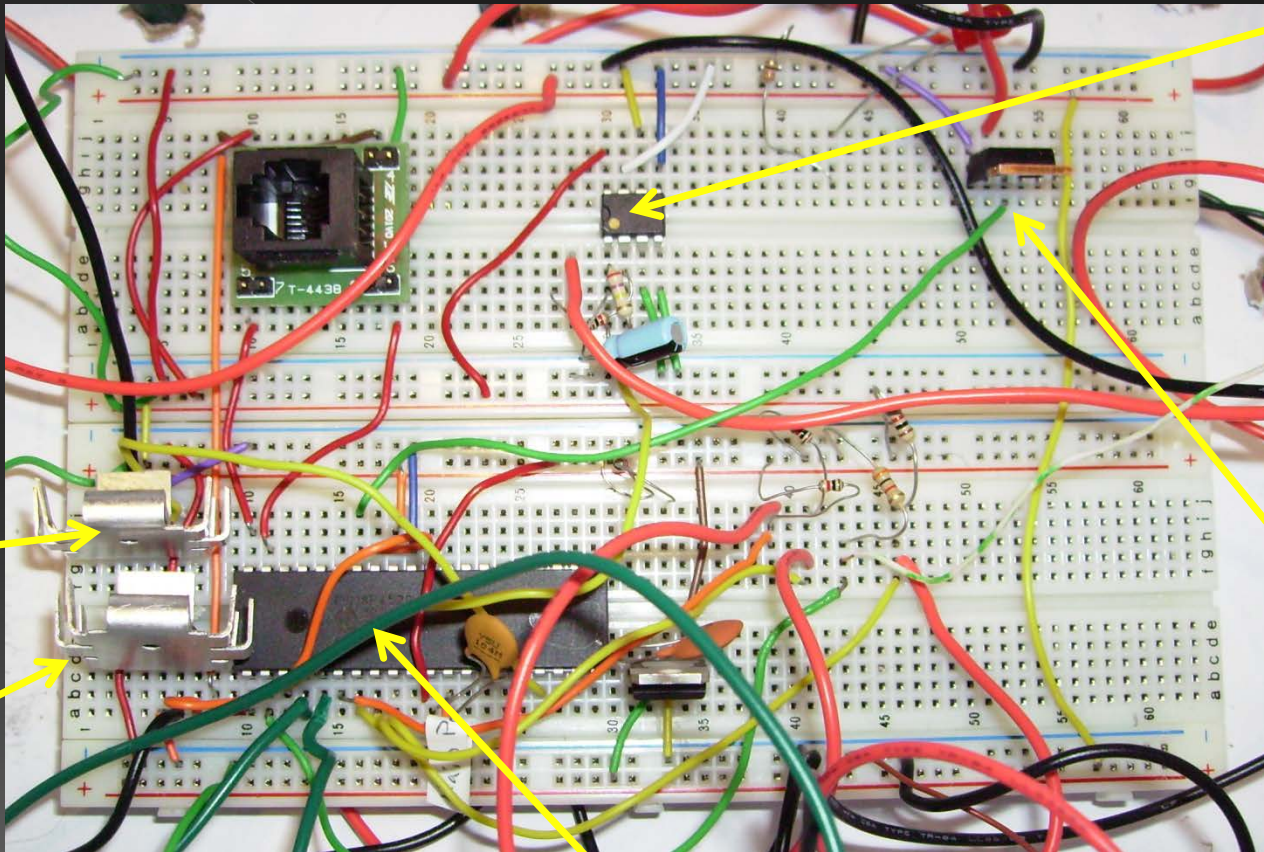
Started with basic circuit outlined in Lab 7

MOSFET for Disco Ball Motor

MOSFET for x-mas lights

Low-Pass circuit for microphone input

MOSFET for flashlight



PIC 18F4520

Additional Components



Microphone



Potentiometers



4.8V motor



Battery Operated
x-mas lights



Switches and CDS sensor



LED flashlight

Our inputs were....

- **All elements on/ All elements off (enable)**
 - Controlled by main power switch
- **X-mas lights only**
 - Controlled by main override switch and sound threshold detected by microphone
- **Spotlight and disco ball only**
 - Sensed by motor main override switch and CDS sensor

Our inputs were....

- **Potentiometer**

- Control the thresholds for light and sound sensors

- **Ambient light intensity**

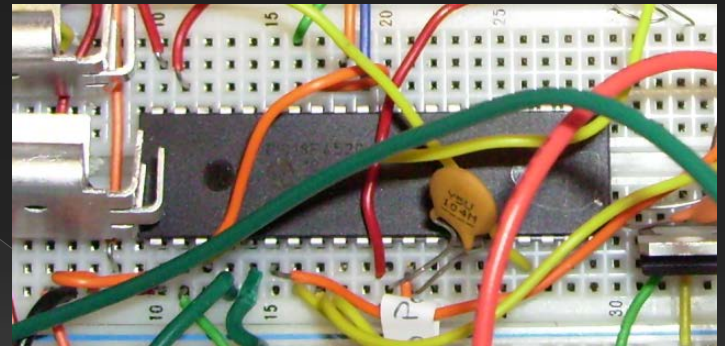
- Sensed using a CDS sensor

- **Ambient sound intensity**

- Sensed using a microphone with an active low pass filter

The PIC as a controller

- We used the following in our code
 - Switches, CDS, microphone, Potentiometers
 - Timer 0 interrupt which ran 200 times/sec and sampled inputs to determine if a change had occurred
 - Spotlight and Disco Ball
 - Used PWM to control the to control the speed of the motor
 - X-mas lights
 - Were controlled solely by parameters within the Timer 0 interrupt



Everything went according to plan!

- Everything functioned as expected



(Click to view video)

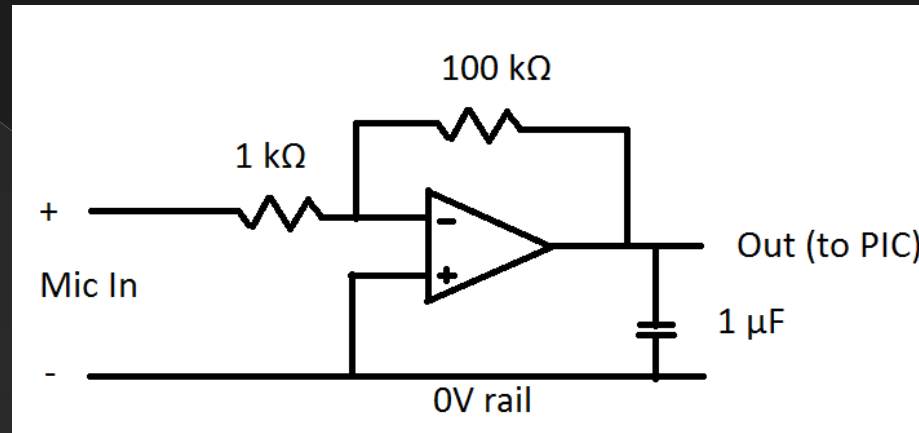
The advice we offer

- It takes a lot of research, time and some EE help to get a microphone to work.
- One bad wiring connection can ruin everything. Sometimes tearing everything apart and starting over is faster than debugging.
- Know your microphones! – They are not created equal some require power but only have two wires.
- Start working early! It takes a lot of time to wire and code any project. Experimenting with trial and error is a time consuming process.

The advice we offer – Sound Input

- ◎ Select a proper microphone
 - Some microphones expect power to run
 - “Dynamic” microphones will work. Anything else is a gamble
- ◎ Circuits that run +V, gnd (the kind we use in mechatronics) only allow reproduction of half the wave.
 - Analysis of sound is difficult, but possible
 - Detection of sound is easy

The advice we offer – Sound input



- Wire OP amp to regulated rail
- Increasing capacitance lowers pass frequency (noise filtering)
- Changing resistance changes gain
- Other circuits work, this is what worked for us

OPERATION: WHACK A MOLE

Tom Hazelrigg

Brian Sutterer

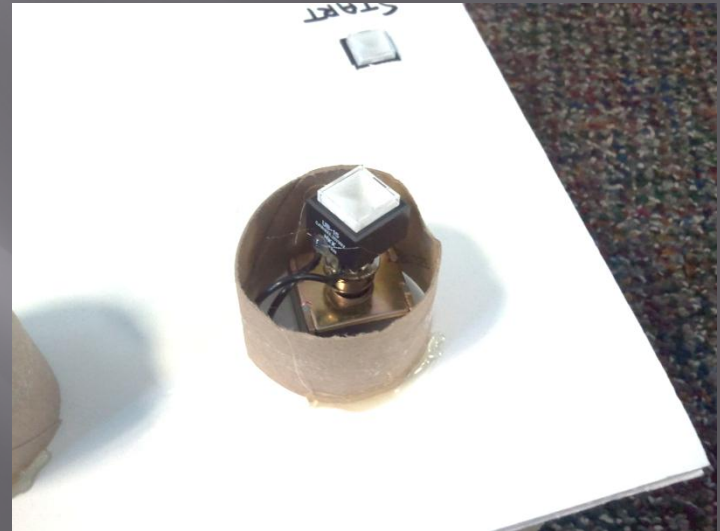
Project Objective

- ▣ Mimic the Classic Whack A Mole arcade game by implementing:
 - 3 Moving moles
 - Sensors to indicate when mole was hit
 - Timer
 - Score counter



Button Inputs

- ▣ All the inputs would be momentary button switches
 - 3 atop each “mole”
 - 1 for Start
 - 1 for Reset

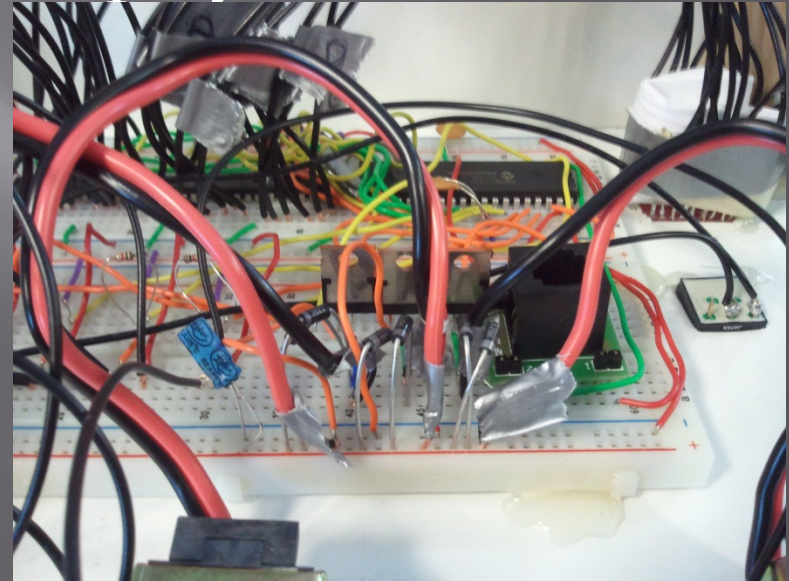


Use of PIC

- ▣ Mainly used interrupts
 - Each mole head was on a dedicated interrupt
 - Timer0 interrupt controlled the output of the timer
- ▣ Everything else was implemented in main()

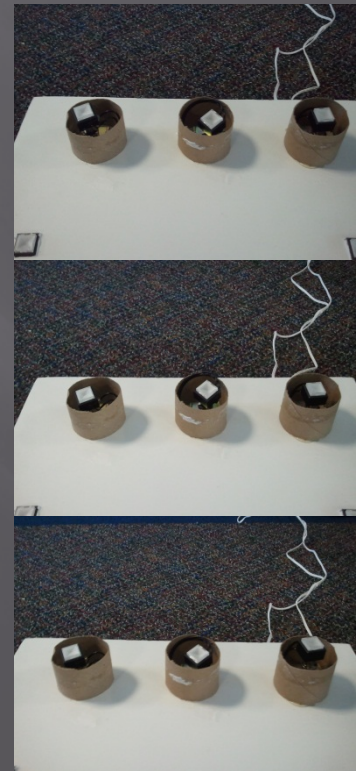
Outputs of the PIC

- ▣ 3 MOSFETs controlled 3 solenoids (moles)
- ▣ 4 seven segment LED Displays
 - 2 for Score
 - 2 for Timer



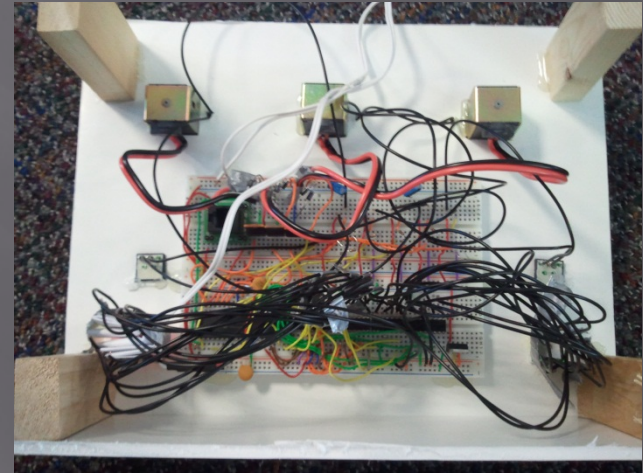
Results

- ▣ Project worked how we said it would (eventually)
 - 3 functioning moles which moved and responded to input
 - Timer display
 - Score display



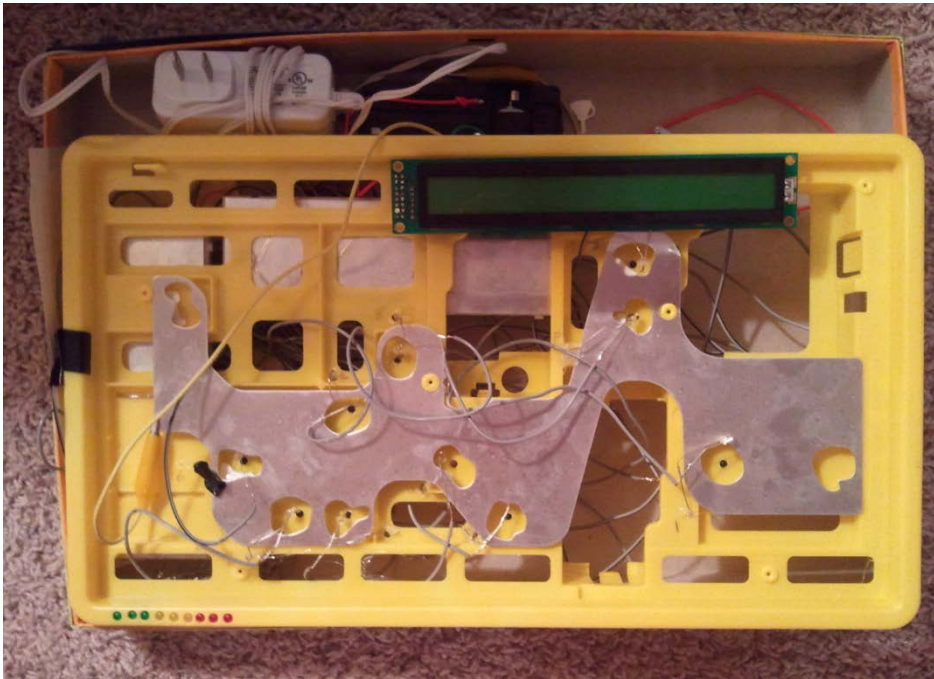
Advice

- ▣ Double check positive/negative ends of power supplies
- ▣ Check for shorts in wiring as you go, not at the end
- ▣ Test small sections of code for functionality at a time
- ▣ Make sure you're buying parts from a reliable source





EMT Training Game: An Update of the Classic Operation Board Game



Aaron Vaslow
Heather Wiest
ME430-02

Project Objective

- 1) Create a game similar to that of the classic game Operation that includes a “patient” with “ailments” that need to be removed.
- 2) Display on an LCD screen vitals for the patient which correspond to specific ailments.
- 3) Detect which ailment the player removed from the game board.
- 4) Update score of the player based on whether the ailment removed was the one that corresponded to the vital.



Inputs

Code:

- ▶ Random number (1–10) to determine the current vital that will correspond to an ailment.
- ▶ Timer that starts when the player turns on the game and ends the game after 3 minutes.

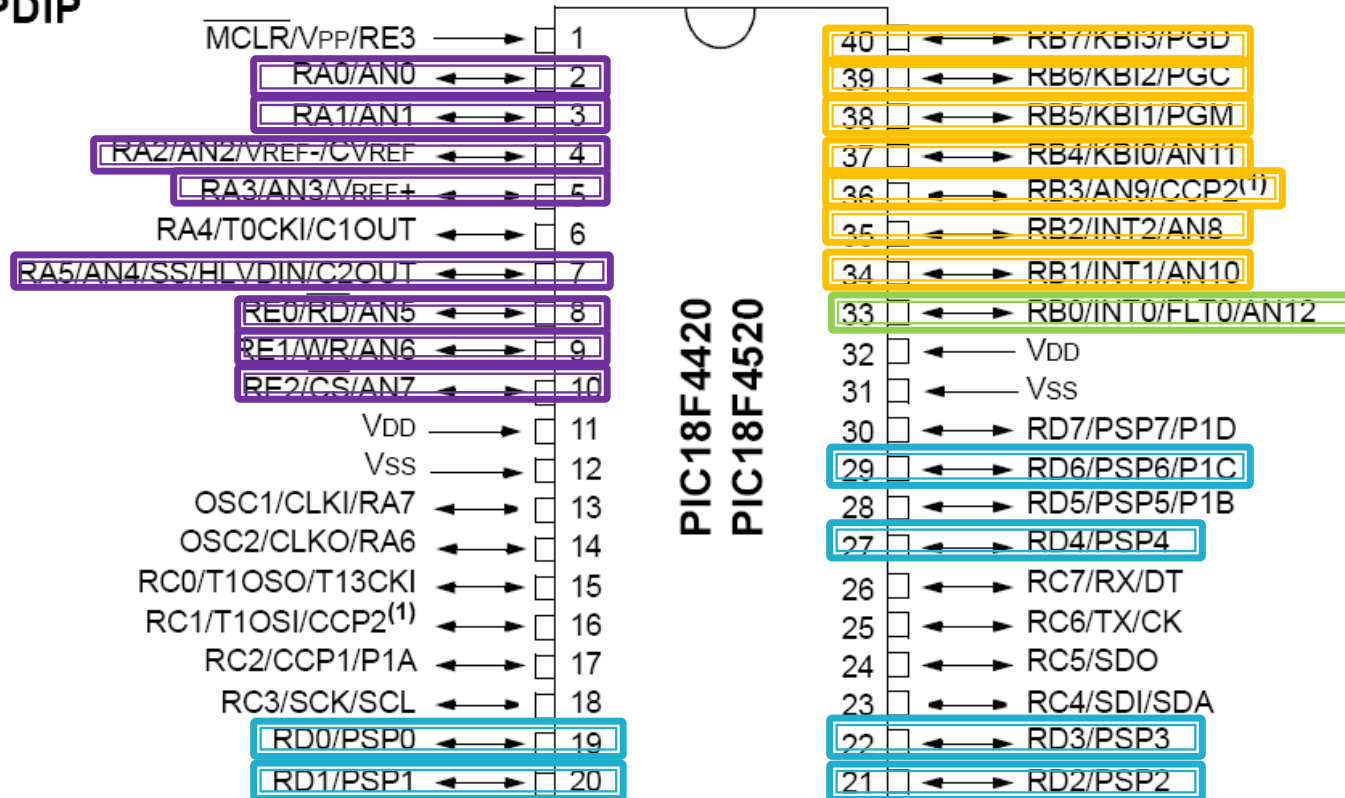
Physical:

- ▶ IR sensors for removal detection.
- ▶ Push button to reset the game.



Use of the PIC

40-pin PDIP



IR Sensors

LCD Display

LEDs

Restart Button

The Code

▶ Interrupts

- Timer0 – record length of game
- Timer1 – check the IR sensors
- RBO – restart button

▶ Logic

- if statements – used to determine if the right ailment was chosen.
- ## ▶ Random Number Generator – used to pick the which vitals would be displayed.

Outputs

- ▶ LCD screen displays vitals for patient and score of player.
- ▶ LEDs light up as the game progresses: green for first minute, yellow during second, and red for the last minute of the game.



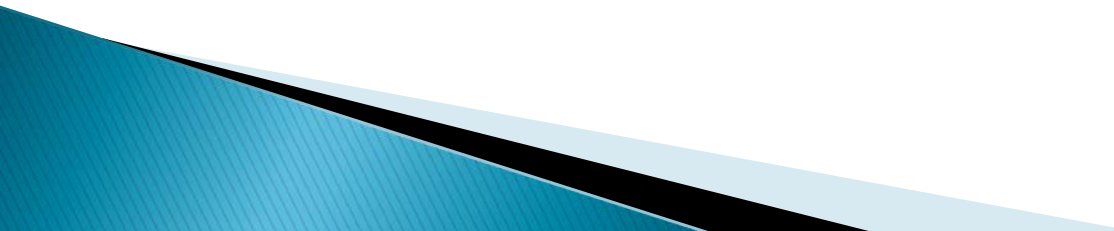
Results

- ▶ Code consistently determined which ailment was removed and updated score correctly.
- ▶ LCD displayed vitals and score.
- ▶ Timer ended game while LEDs lighted up as player ran out of time.

Main Problem: Needed to replace IR sensors multiple times.



Advice

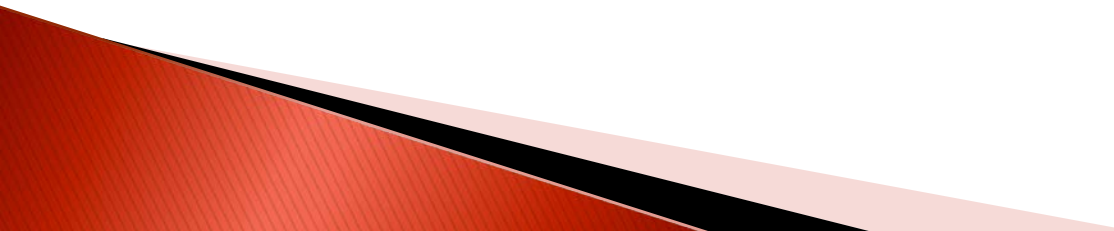
- ▶ Try to keep your project SIMPLE.
 - ▶ Start EARLY to allow for ample debugging time.
 - ▶ Fully debug your code before trying it on your final apparatus if there is complex wiring.
- 

Field Goal Challenge

Robert Alumbaugh
Drake Sayre



Objective

- ▶ Create a paper football game which detects if a field goal was made
 - ▶ Randomizes the placement of the kicks by lighting up LEDs
 - ▶ Uses a timer as a game clock
 - ▶ Displays the scores for 2 players on seven segment displays
- 

Inputs

- ▶ IR sensor detects if a field goal was made
Made: Breaks the light beam
Missed: Detector sees no change
- ▶ Push buttons initiate the start of the game for each player

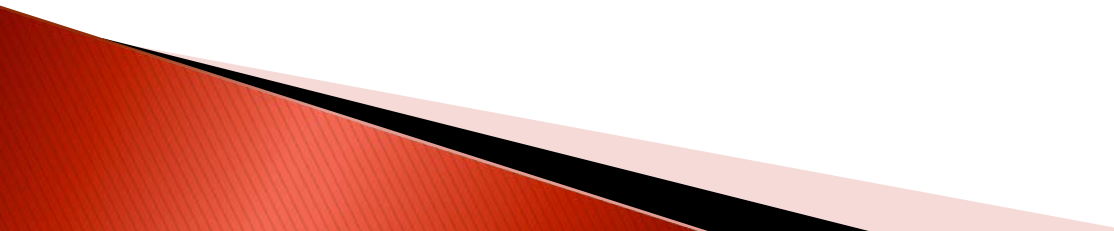
Using the PIC

- ▶ Used interrupts and timers for randomizing the LEDs
- ▶ Used delays for the game clock and for a brief pause before each players game clock started
- ▶ Used an ADC conversion for the IR emitter/detector to detect if a goal was made

Outputs

- ▶ The players score was incremented depending on the ADC result and displayed using seven segment displays
- ▶ The game clock utilized delays to count down from 5 seconds to 0 seconds at each LED position

Results

- ▶ Everything worked as planned except for small IR emitter/detector issues
 - ▶ The distance between the emitter and detector was too far apart which forced us to set the threshold value high
 - ▶ The result of this was that sometime a goal was not detected
- 

Advice

- ▶ Try to use a better quality emitter and detector so that each goal will accurately be counted
- ▶ Using one PIC for 5 seven segment displays can be tough (maybe use 2 PICs)

Forget-Me Not

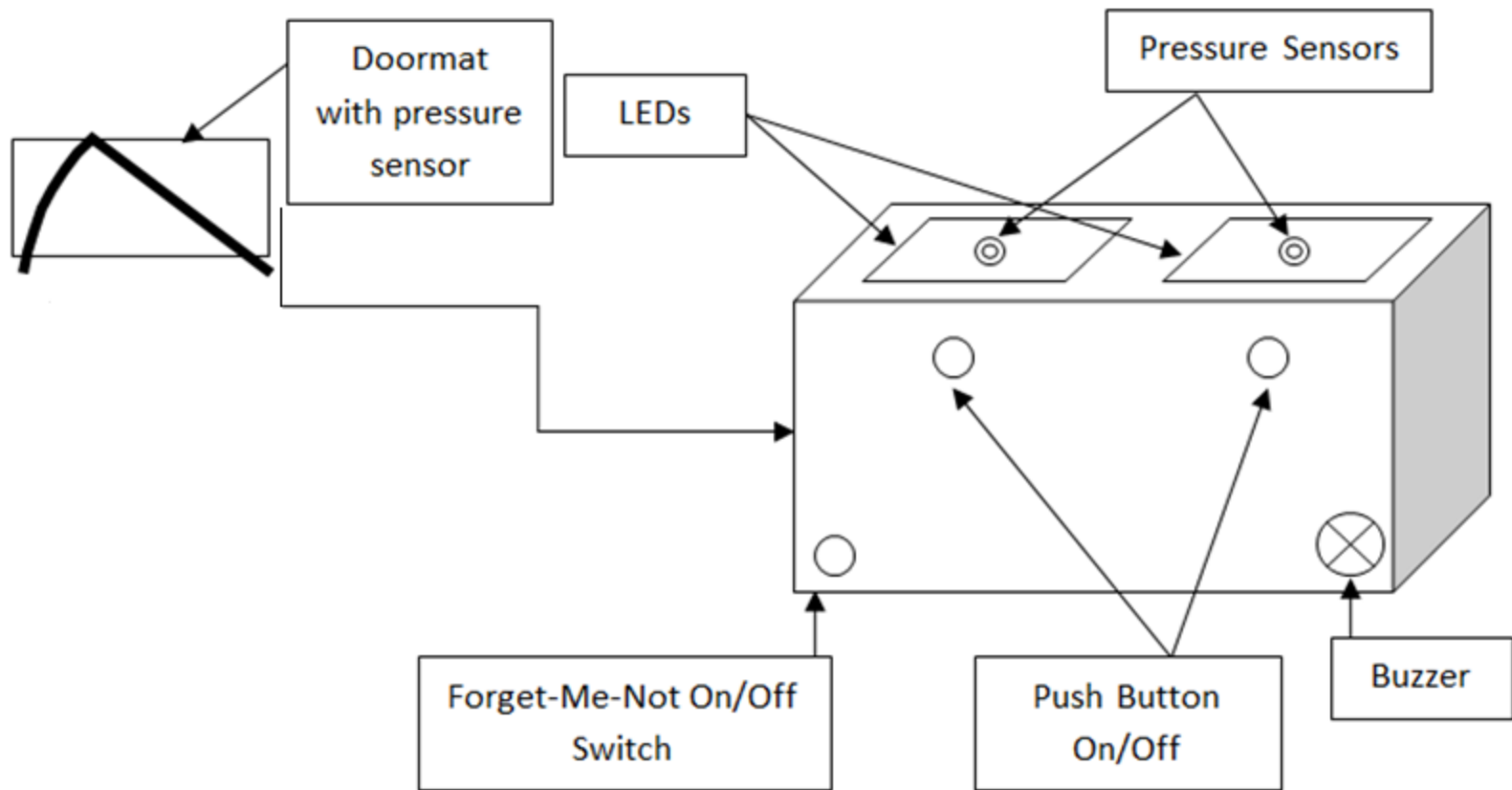
Megan Chann

Ryan Shaffer

Project Objective

- **Problem:** Many people regularly forget items like their phone, wallet, or keys at their house
- **Solution:** create a device that allows you to set important items on it at night, and in the morning when you go to leave your house the device will alert you if the items have not yet been picked up.

Project Overview



Inputs and Outputs

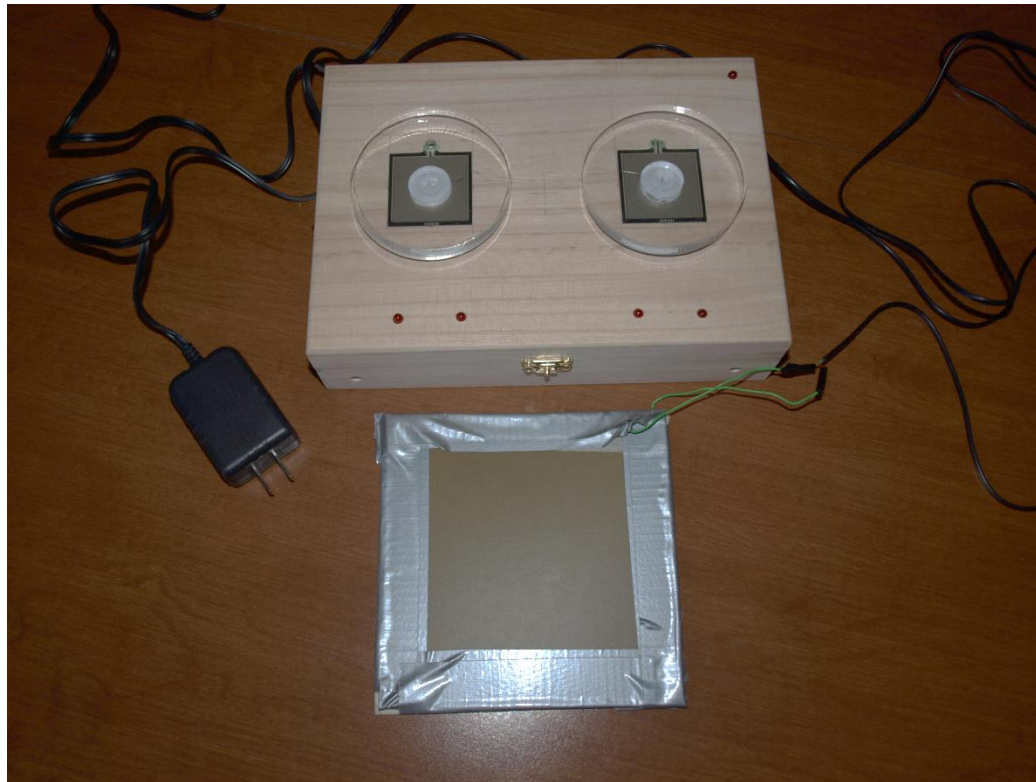
Inputs	Sensor
Slots turned on/off	Position switch
Items in slots	Force sensitive resistor
Doormat stepped on	Analog sensor
Device on/off	Position switch
Outputs	Indicator
Power on	LED on
Slots on	LED on
Slots active	LED on
Alarm on	Piezo Buzzer

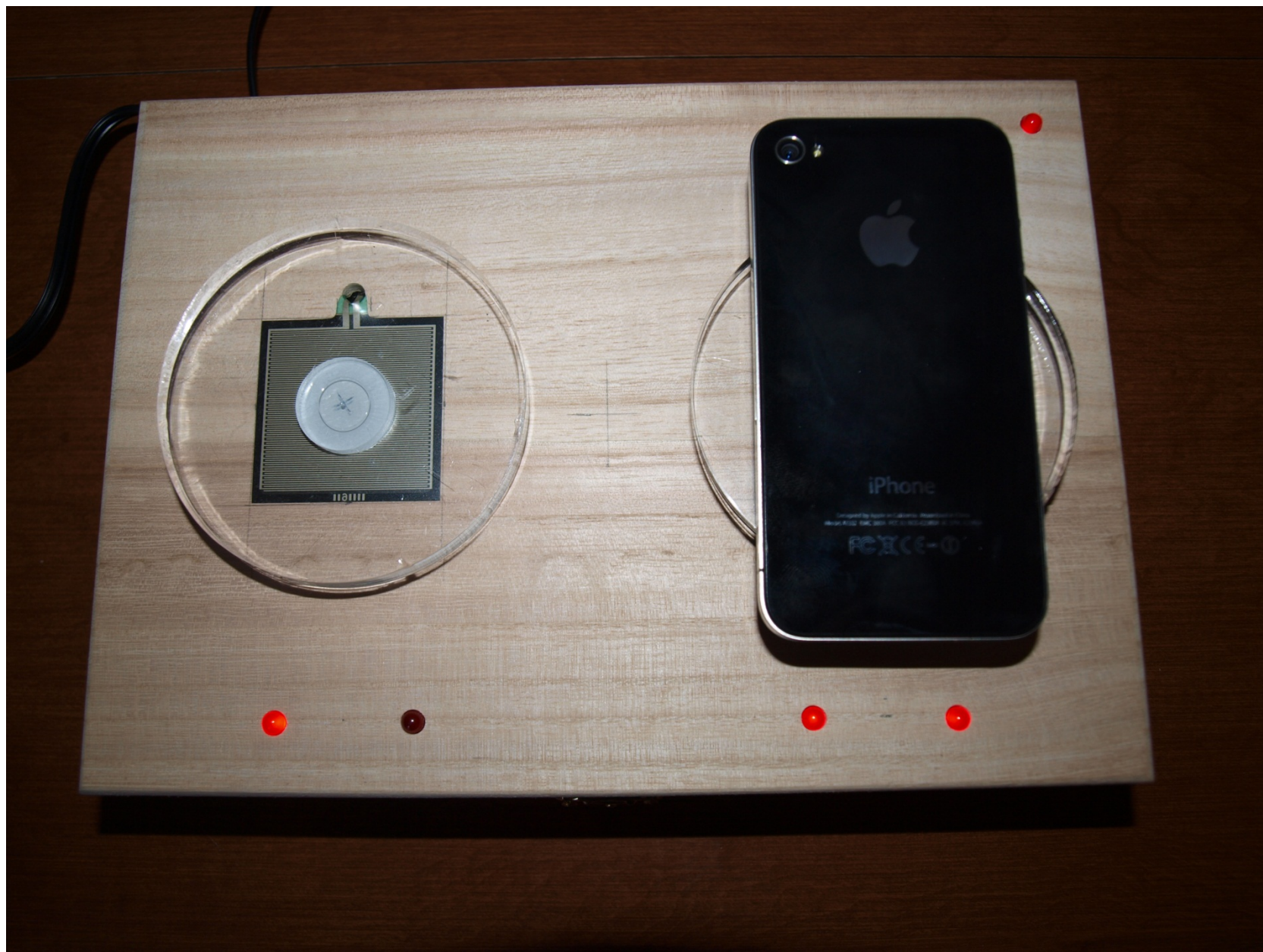
PIC Control

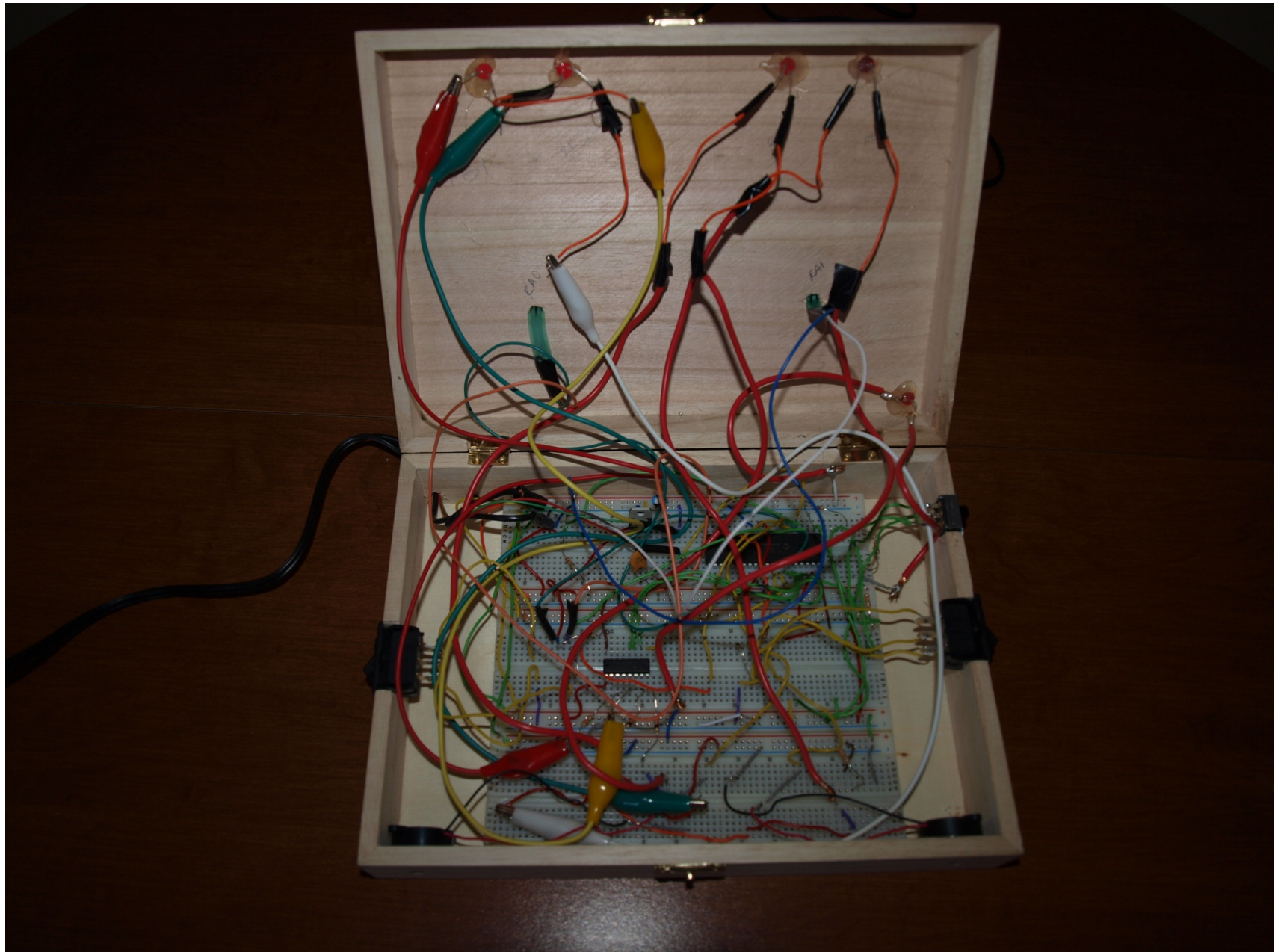
- All code written in main loop
 - Position sensors all coded using recent button states to indicate position
 - Doormat sensor consisted of two sheets of copper used as an analog sensor
 - Force sensitive resistors read using ADC
 - Piezo buzzers run using PWM

Results

- The project was fully functional and matched the original proposal







Advice

- Start your project early
- Wire the project in stages because that makes it easier to debug
- A mechanically simple project will take you much less time to put together and will end up working much more consistently
- Don't underestimate the amount of time it will take you to solder and build your project

Haptic Compass



Kim Murray and Phil Scherer
ME430 Project for Dr. Olson

Project Objectives

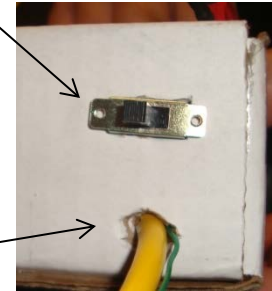
- Create a belt that will use magnetic fields to determine North
- Activate motors to show user which direction to turn to face North.
- Listen to direction commands from computer

Inputs

- On/Off switch for powering the device
- Magnetic field readings to HMC6352 (compass chip)
- UART from computer

On/Off switch
outside the box

9 wire
outputs



Note:

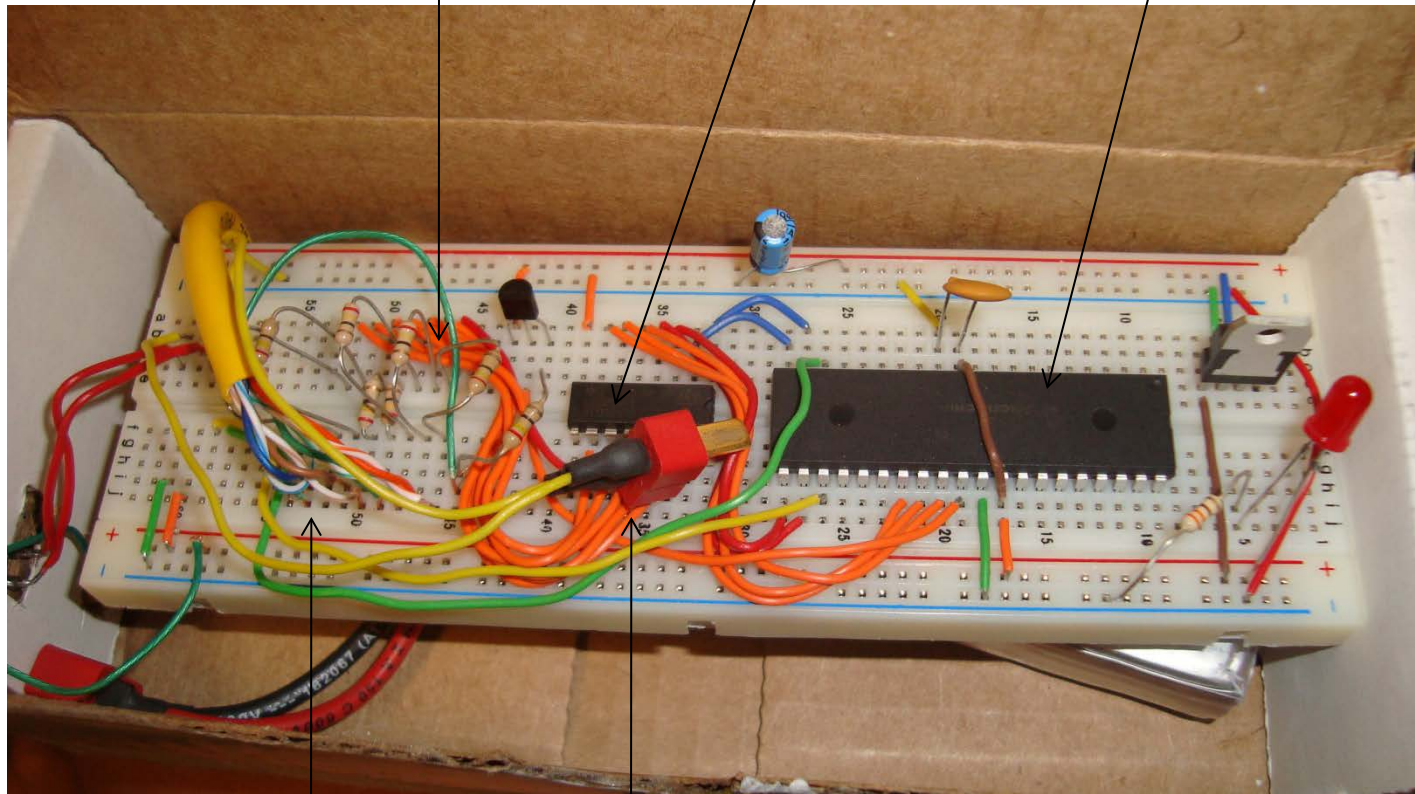
The compass chip was not included in the setup since we were unable to get it to function

PIC as the controller

8 resistors between
Darlington and output

Darlington

PIC program to control
motor outputs

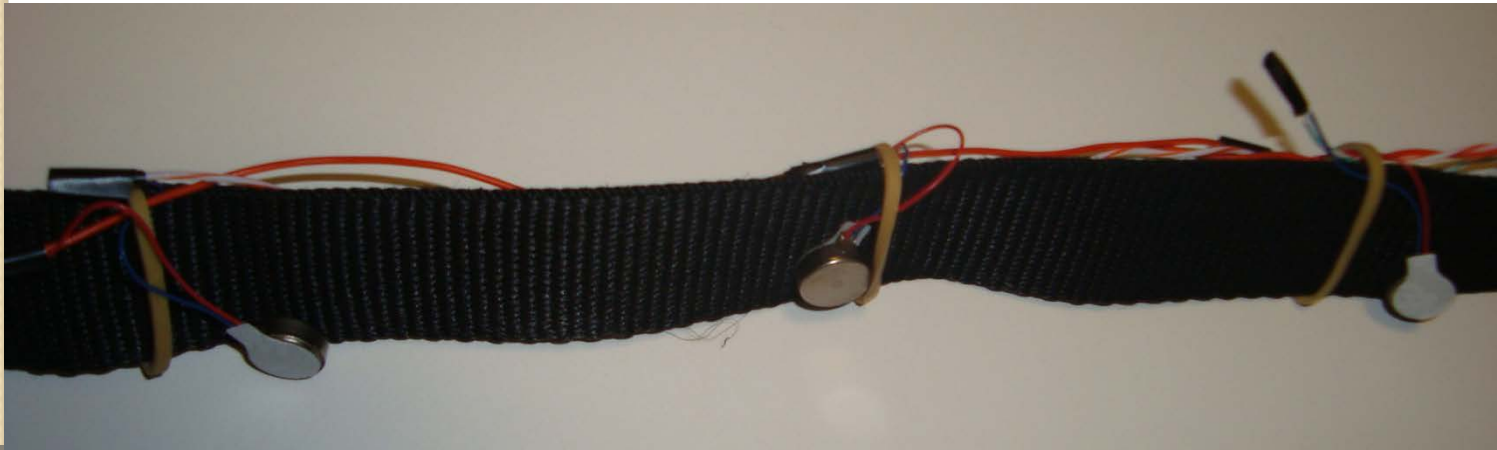


Leads to the motors on the belt

Connector to battery pack, makes
system mobile

Outputs

- 8 small vibrating motors attached to a belt
- One lead connected to 5V regulated power, the other to the breadboard with the resistor between the Darlington and the lead.



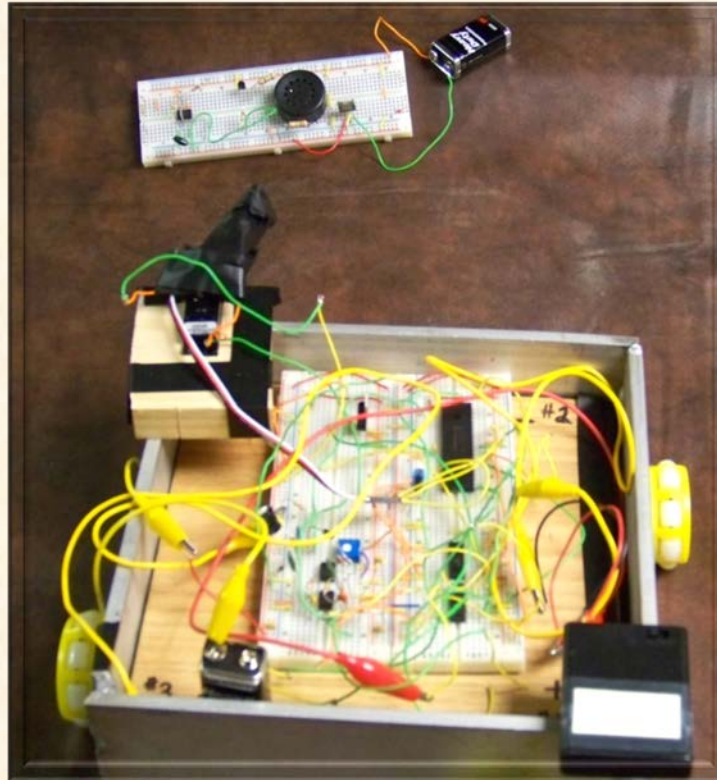
Results

- We were unable to get the compass chip to work correctly
- We used a demo program that worked and cycled through what the motors would potentially do if the compass chip had worked

Advice

- Read spec sheets and data sheets provided by manufacturers before purchasing and ensure the PIC can handle all the necessary protocols
- Start early! It will help if you need extra time to learn about something not covered in depth in the class.

Omni-Wheel: The All Directional



Steven Harden

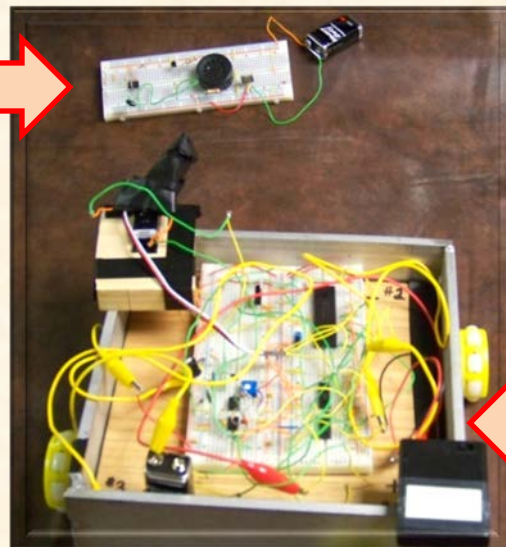
Taylor Coleman

Mechatronics – ME430-02
Instructor: Dr. L.G. Olson

Project Objective

- # Create an Omni-directional robot capable of sensing a given noise frequency, be able to move toward the location of the noise, and be able to follow the noise if it changes location.

Noise Source



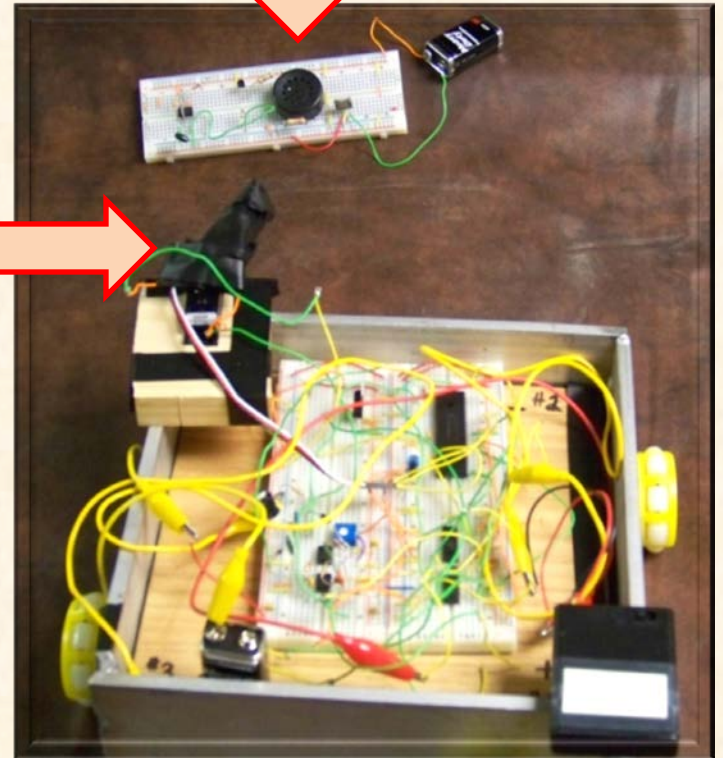
Omni-Directional
Robot

Inputs to PIC

1200 Hz Sound from
Speaker Detected by the
Filtering Microphone

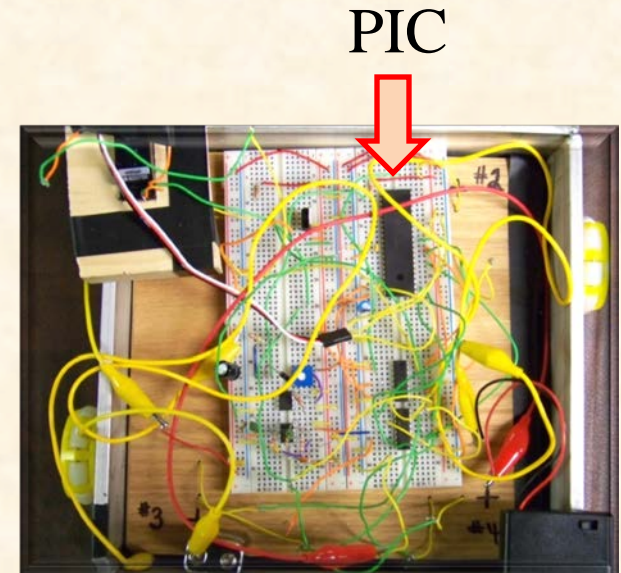
Microphone
Housing

Speaker



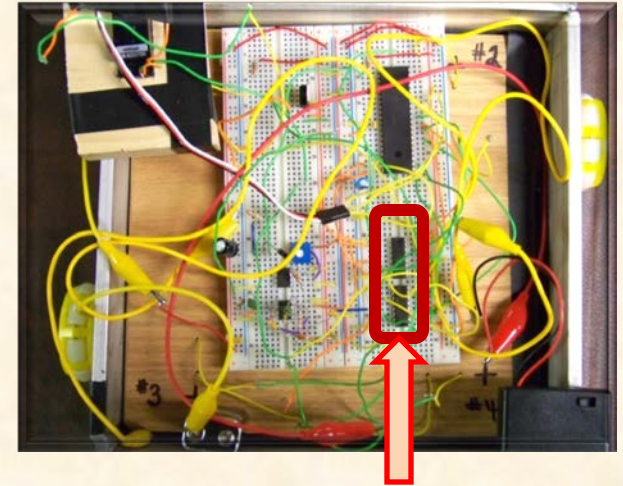
PIC Programming

- # Only one PIC was used
- # The entire system is continuous, so all logic is in an infinite while loop
- # 1200 Hz frequency was converted using ADC through the microphone
- # The microphone was placed on a servo motor which constantly scanned the environment to locate the 1200 Hz noise source
- # Delays and PWM were used to control the each motor to move the robot in the direction of the noise



Outputs

- # The PIC was used to send signals to two H-bridge chips, providing bi-directional motion on all four motors on the robot
- # When the robot was turned on and a 1200 Hz noise was not detected, the robot would remain stationary, but would constantly scan the environment for the noise
- # Once the 1200 Hz noise was detected the robot would move toward that noise location



2 H-Bridge Chips

Results

Successful Areas

- ✦ The robot was able to move from a stationary state with no noise detected to a moving state when a 1200 Hz sound was sensed
- ✦ Omni-directional motion of the robot was attained with the use of all four motors

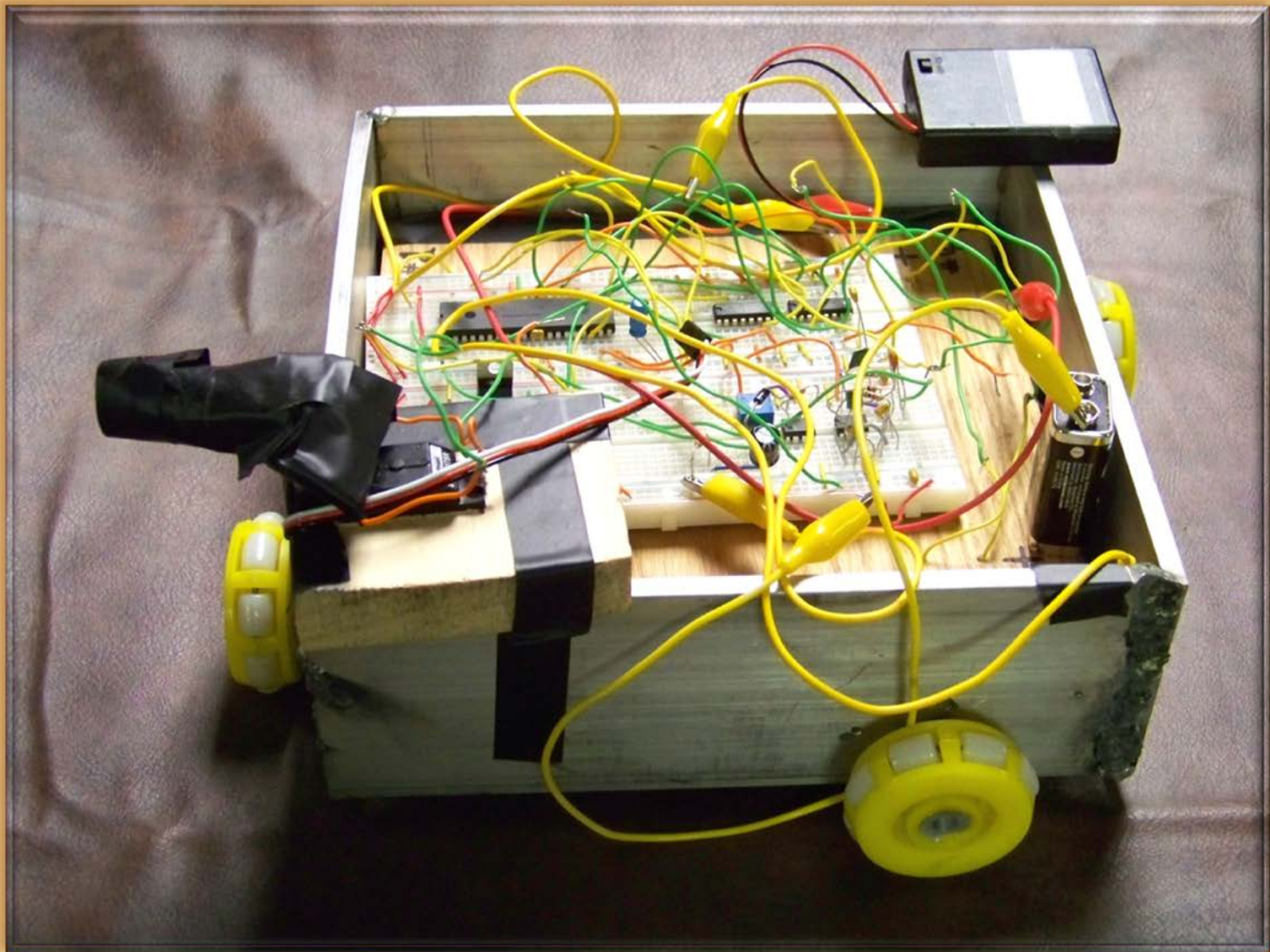
Problem Areas

- ✦ The robot was unable to accurately follow the noise source

Therefore, our project did not work as we had originally planned, but we were able to get most of the aspects of it functioning properly.

Advice to Future Generations

- ✚ Building a directional microphone is a lot more difficult than some may think, it might be best to just purchase one rather than building it
- ✚ When purchasing a Servo motor, make sure it is NOT a continuous Servo and is similar to the ones used in the in-class labs
- ✚ Make sure you know how many PIC pins your project will require before you start building and draw up a plan for the circuit ahead of time
- ✚ Carefully check all lines of code when programming the PIC so that each partner knows what is going on and to minimize any future debugging issues



Improving the Game of Cornhole

By

Charles Biggs, Ryan Elliott,
Alex Pantages

Objective

- Our main objective was to make playing the game of Cornhole exciting and more fun to play
- We wanted to make a scoreboard for the players to use
- We wanted there to be flashing lights and a buzzer that goes off when a bag was thrown through the hole

The Inputs

- We had two types of inputs; IR Sensors, and Push Buttons
- We placed the IR Sensors inside the hole that would create an IR beam that will be broken when a bag is thrown through the hole
- We had 5 simple Push Buttons on the scoreboard; a reset button, a plus button and subtract button for each team

Using the PIC

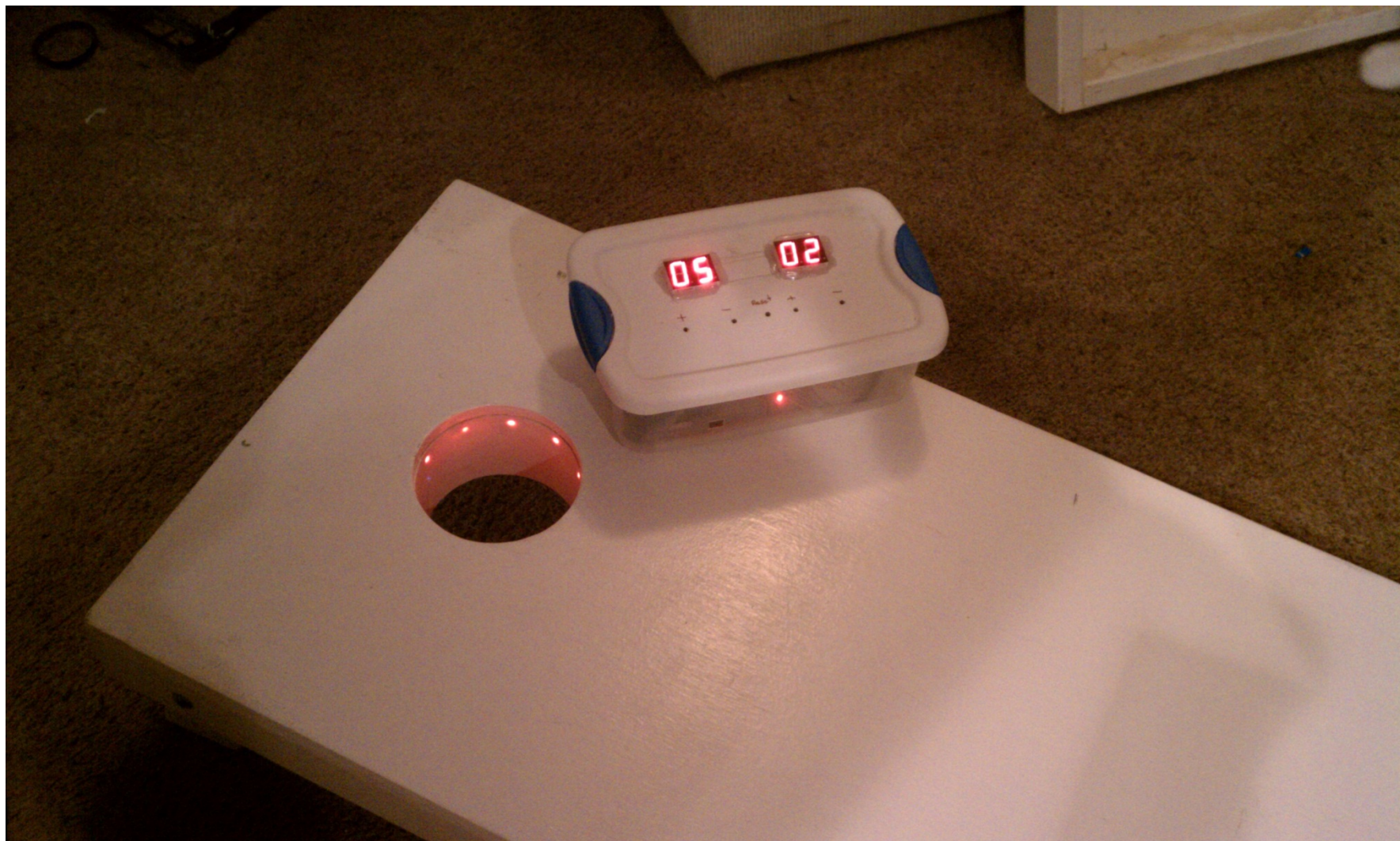
- The PIC acted as the brains behind our project it used polling to continually look to see if the bag had gone through the hole
- We used PWM for the buzzers, ADC for the IR detector, timers, and delays to allow the buzzers and LEDs to flash
- We did not use any interrupts

The Outputs

- We had three types of outputs; Piezo Buzzers, Flashing LEDs, and 7 Segment Displays
- When the IR beam is broken by the bag, the LEDs are set to flash and the Piezo Buzzer is set to sound
- When the Push Buttons are pressed, they either added a point, subtracted a point or reset the 7 segment displays that made up the scoreboard.

Results

- The project did end up working like we wanted it to
- Every time the bag goes through the hole, the buzzers sound and the LEDs flash
- The score board easily allows the players to keep track of the score
- We think that our new boards and the scoreboard will make Cornhole a lot more fun to play



Advice

- We were disappointed in the sound of our buzzers. In order for the buzzer to be loud enough we had to use a really annoying frequency so find a loud enough buzzer that will allow you to choose a nicer frequency
- Another thing that we would have liked to do is integrate the scoreboard into one of the boards but we felt that we would have needed a lot more time for that so we would have liked to start a little sooner on the project

PIR Alarm System

Brad Jeffers

Alex Titus

2/22/11

ME430-02

Project Objective



Create an alarm system utilizing a passive infrared (PIR) sensor to detect intruders, a buzzer to play an alert tone, as well as an LCD for displaying current alarm state.

Environmental Inputs

- * Passive Infrared Sensor (PIR)
 - * Creates snapshot of room upon power-up and updates every half-second
 - * Sensor outputs low signal if different snapshot is seen
 - * Uses analog pin on PIC
 - * Sensitivity controlled via potentiometer
- * Push Button
 - * Momentary push button to reset alarm



PIC & Programming

- * The logic for the alarm system was created using a switch() statement containing 3 cases (0 through 2)
 - * Case 0 – System begins countdown which is displayed on LCD as well as a flashing LED. After countdown finishes, code switches to case 1.
 - * Case 1 – System is armed and waits until the sensor ADC registers above the coded threshold. When the sensor goes low the LCD displays “INTRUDER ALERT!”, the buzzer sounds, and the code switches to case 2.
 - * Case 2 – System is in intruder mode and waits until reset button is pressed which causes the buzzer to be turned off and the code to switch to case 0.

Outputs

- * LED

- * Used to indicate current state of device
 - * Blinking for arming stage
 - * Off for armed stage
 - * On for intruder stage



- * LCD Screen

- * Sends one of the following messages depending on state:
 - * “Arming...(seconds left on countdown)”
 - * “Armed”
 - * “Intruder Alert!”



- * Buzzer

- * Emits sound when alarm is tripped



Results

- * The device worked well to our expectations
- * The sensor had great range (~20ft) and we were able to control its sensitivity
- * The project was and easy to build yet very effective for its purpose (\$40 total)
- * Multiple buzzers should have been used to increase effectiveness of alarm
- * Toggle switch should have been implemented to control power to device

Advice

- * Do not make your project overly complicated, you will be surprised how difficult simple tasks are to implement
- * Start as early as you can so that you have the opportunity to flourish ideas further into the project
- * Labs consume all of class time and spill over into out-of-class work. Work on your project concurrently with labs to avoid a lackluster project

Joystick Controlled Paintball Gun Turret

ME 430 Mechatronics

Anthony Millhouse

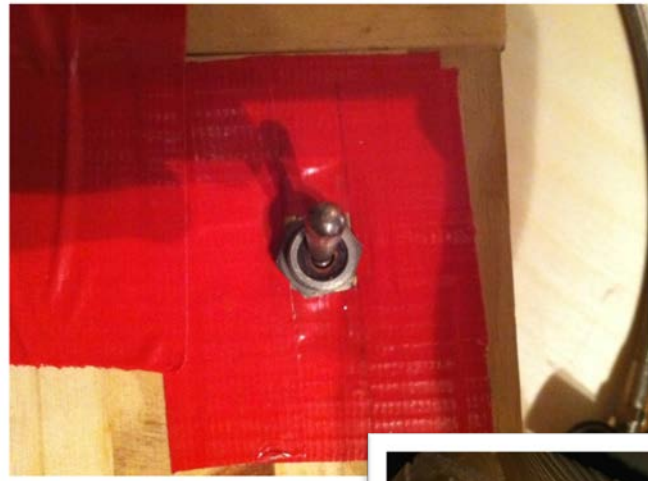
Christopher Duncan



Objective

Create a paintball gun turret which can be controlled by a joystick

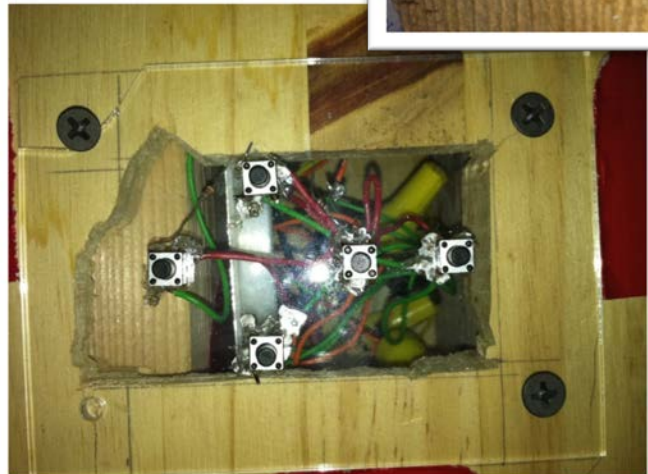
7 Inputs



On/Off Switch



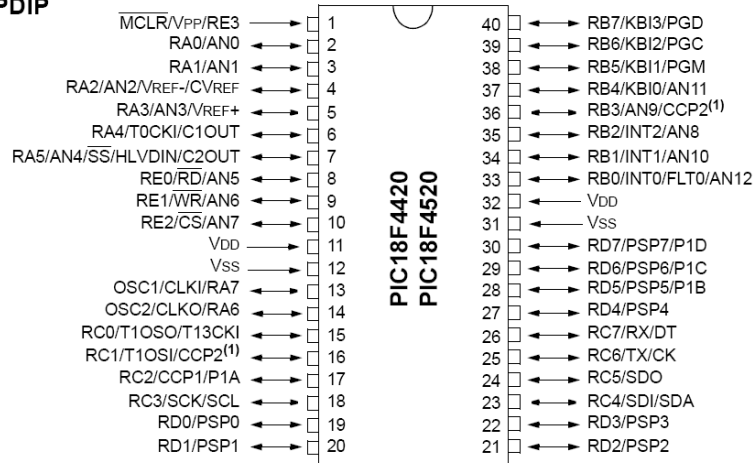
Potentiometer



“Joystick”

Switch:
Up, Down, Left, Right, Fire

40-pin PDIP



PIC Microcontroller H Bridge

- Used the PIC to control inputs for the potentiometer, left/right from joystick, on/off switch, and fire switch
- Used an H Bridge to make the motor in the front bidirectional
- Used a timer interrupt. Once in interrupt mode, the program would test each statement. The servo motors were controlled by delays. PWM was not used for the DC motor. Analog to digital conversion was needed for the potentiometer



Trigger Servo Motor “Fire”

Stayed in neutral non-fire position and then would move over to hit the trigger when the switch was pressed

3 outputs

Base Servo Motor “Left/Right”

Servo would move to 0 degrees when the “right” switch was pressed and move to 180 degrees when the “left” switch was pressed



Front Gearhead DC Motor “Up/Down”

DC motor would spin in one direction when the “up” switch was pressed and switch directions when the “down” switch was pressed



- Stripped the base servo connection for left/right control so the turret gun only detected its up down location via the potentiometer, shot, and moved up and down.

Results

- Have additional PIC chips in working order in case chips are fried late in the project
- Check all of the connections in the wiring including connections associated with the breadboard as some may come loose
- Read manufacturer specification sheets to check allowable current and voltage
- Label all wires with an appropriate tag to save time and prevent misconnections
- Have all components of the project working together before attempting to put elements in a presentable box or mounting location.
- Be cautious when attempting cut a gear to fit onto the end of a motor out of plastic; gear teeth may become stripped.
- Start the project early and even plan on having it completely finished a week ahead of time; it can be surprising how many new problems can arise at the end of the project

Advice

Laser Race



Cale Snyder

Steve Jones

Objectives

- Create a two player racing game modeled after the carnival squirt gun race
- Utilize skills learned in ME 470 to design and build game from the ground up
- Have fun!

Inputs

- CdS Cells: Change resistance under different light levels. These are used to sense whether a players laser is hitting the target.
- Potentiometer: Used to set the threshold for the CdS cells. Both the potentiometer and the CdS cells report an analog signal to the PIC
- Pushbutton: Simple switch circuit used to reset the game when button is pushed

PIC

- ADC: We used the ADC to convert analog signals from both the potentiometer and the CdS cells to a number between 0 and 1023.
- PIC continuously checking whether or not CdS signal value is higher than potentiometer signal value.
- If CdS cell is being hit with laser, stepper motor takes one step before the PIC again checks the CdS cell signal value
- Delay used after each step of stepper motor to control speed.
- Counting variable keeps track of number of steps each motor has taken, certain number of steps signals end of race.
- At end of race, code goes into a loop in which all game function stops (motors will no longer run if target is hit), and an LED signifies the winner.
- If pushbutton is pushed, all variables reset to initial values.

Outputs

- Stepper Motors: Controlled using Pic signals and and H-Bridge. The stepper motors turned if their corresponding CdS cell was being hit with a laser, pulling a monster truck along the deck.
- LED's: Two LED's are located, one by each target, to signify which truck is the winner at the end of the race.

Breadboard

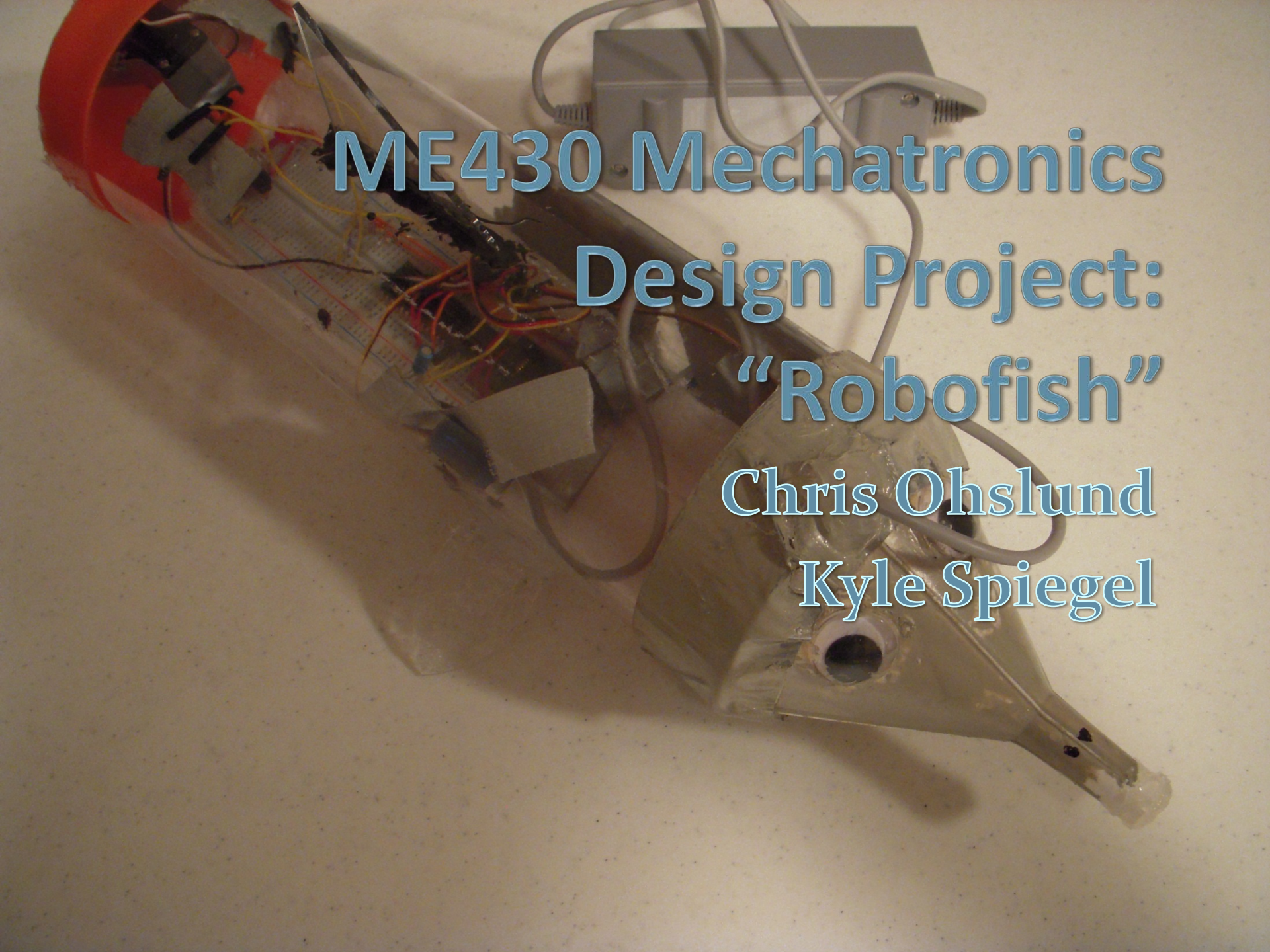


Results

- The game not only works, but is fun!
- Initial stepper motors had to be switched to lower powered motors due to the current requirements of running both motors at the same time. New motors worked great.

Advice

- Only purchase parts with known specifications
- Code in chunks, or modules!
- Order parts early so you have time to experiment with them and make changes if necessary. This helped us to overcome a problem with the shipping of our parts, still having plenty of time to build our race.

The image shows a custom-built robotic fish, 'Robofish', resting on a light-colored surface. The fish's body is constructed from a clear, cylindrical plastic bottle. Inside the bottle, a breadboard is populated with various electronic components, including integrated circuits, resistors, and jumper wires in red, yellow, and black. A small DC motor is mounted on the breadboard, connected to a grey power supply unit with a USB-A to USB-B cable. The fish's head is a clear plastic cone with a circular opening for a camera lens. A red circular cap is visible at the rear of the bottle. The text 'ME430 Mechatronics Design Project: "Robofish"' is overlaid in a blue, serif font with a drop shadow effect.

ME430 Mechatronics Design Project: "Robofish"

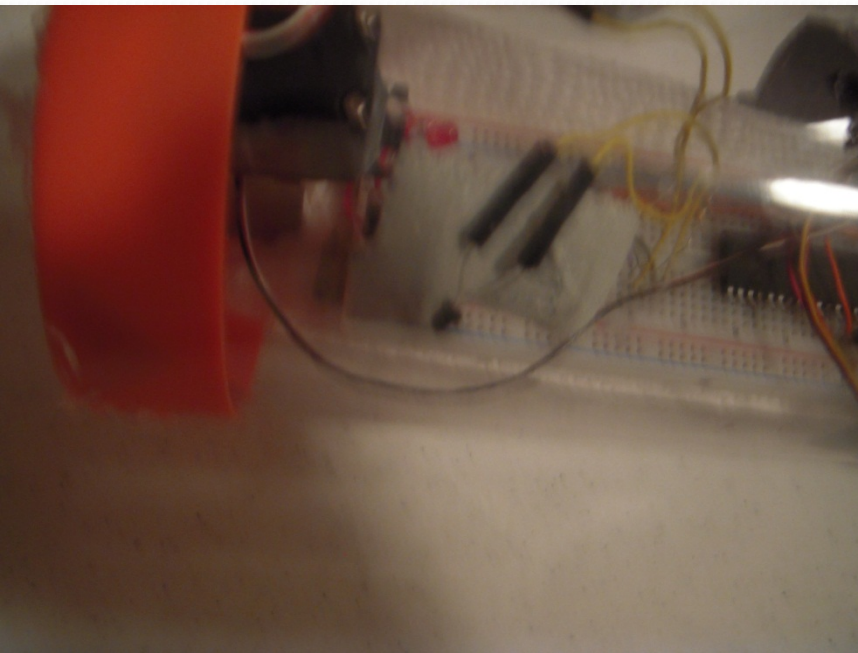
Chris Ohslund
Kyle Spiegel

Project Goals

- Make a fish!
 - Very important: Needs to be waterproof
 - Needs to be mobile
 - Needs to look the part
- Make it a robot!
 - Capable of predetermined motion
 - Controllable by external input

Controlling the 'bot

- Done through infrared sensing
 - Robot is built around a thin polypropylene hull – transparent to IR, and very waterproof
 - Plan: Wave an IR source at the bot, and have it react



- PIC would interpret this as analog signal
- Use ADC to get a useful digital signal

Fig. 2: Infrared sensor.

Moving the 'bot

- Servos on the fins!
 - Servos are simple, and offer high torque in a controllable range
 - Important to make sure they don't go too far
 - Might break water seal
- Controlled via the PIC
 - Servos require a pulse; we used interrupts to produce this
 - More reliable than PWM

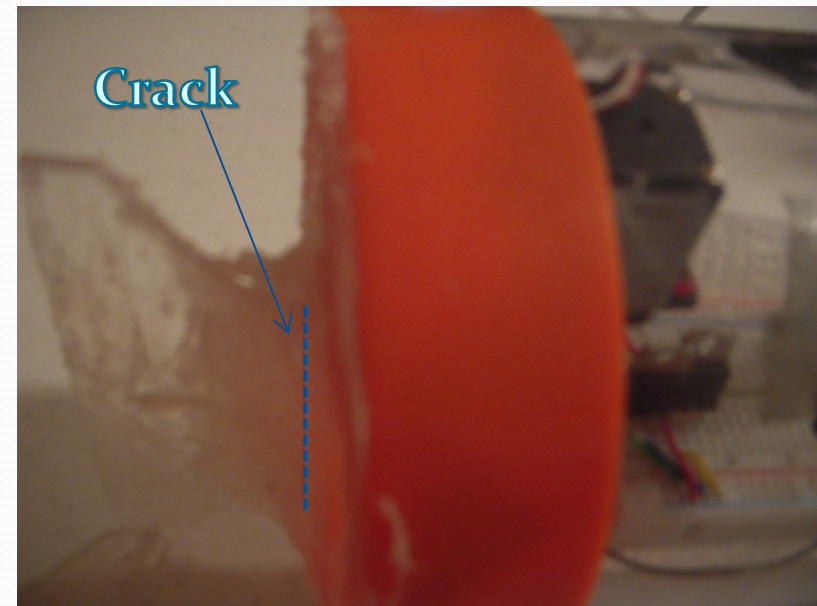
Fig. 3: Motor & fin assembly



Results

- Turns out acrylic (which we used for our fins) is actually pretty fragile
- We dropped the robot on its tail fin the day before demonstration, which disabled our robot's sole means of generating forward motion
 - Able to correct one major problem – the need to reseal the tail – by adding more silicone around cracked portion & drying rapidly
 - Could still drift forward by rapidly fluttering side fins

Fig. 4: Rear fin assembly



Lessons

- Don't drop the project!
- Make or buy many, many backups of any part that is cheap but inconvenient to replace
 - We could have made two dozen more tail fins with the acrylic we still had, if we had had enough time to do so
- Check your parts often when they are plugged in!
 - We slagged a battery pack; this is something to avoid
- Find a way to avoid waterproofing a part if possible
 - They make waterproof servos for a reason; waterproofing is difficult to do, and water-sealed parts won't perform as well

ME 430-Mechatronics Pinball Machine

By:

Luke Crites

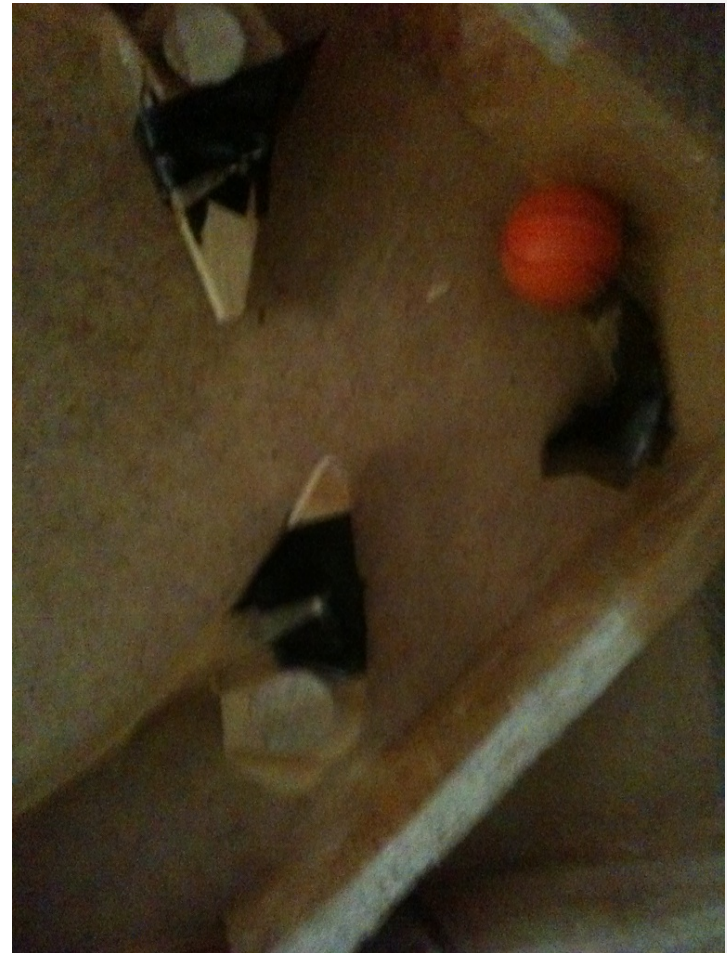
Issac Stockrahm

Project Objective

- The overall objective of the project was to design and build a functioning pinball machine.
- It would have mechanical paddles to propel the pinball.
- Score would be kept whenever the ball triggered a switch, which was then recorded on 3 7-segment displays.
- When the ball falls between the two paddles a separate switch would trigger a ball counter, recorded on another 7-segment display that once it hit 3 would end the game.

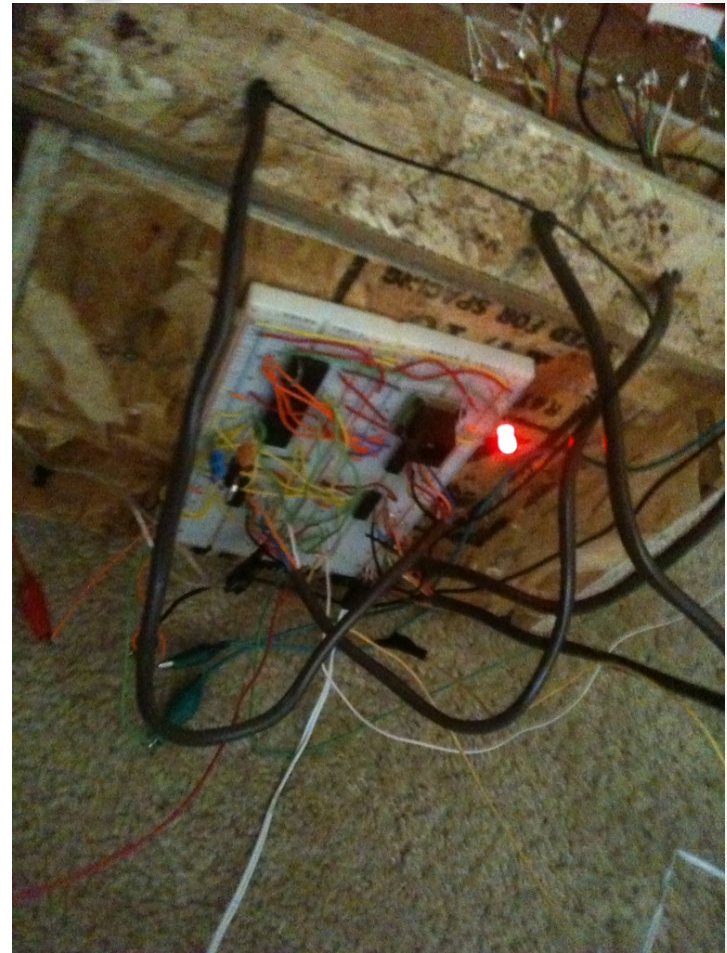
Inputs

- Because of a limited supply we would only use 2 lever switches to score and a third for the lost ball counter
- Once a scoring switch was triggered a signal was sent to increase the scoring 7-segments, and the same for the lost ball switch
- Turning the game on was controlled by plugging in the power source



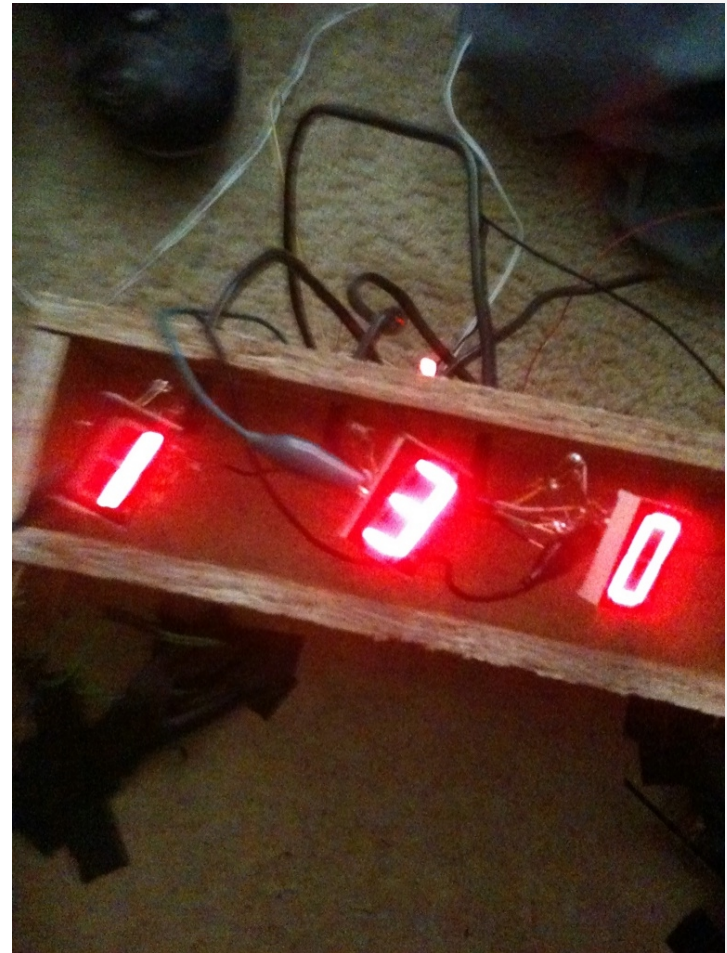
PIC Development

- The entire game was ran off a single PIC
- Interrupts were used to program the PIC and get our desired outputs
- It controlled a few delays that controlled our lost ball counter
- It also controlled our 7-segment displays



Outputs

- Once a switch was hit signal was sent to the 7-segment displays, increasing the score by 20 each time
- Also with the same was done when the lost ball switch was hit
- Finally when the lost ball counter hit 3 the entire game reset



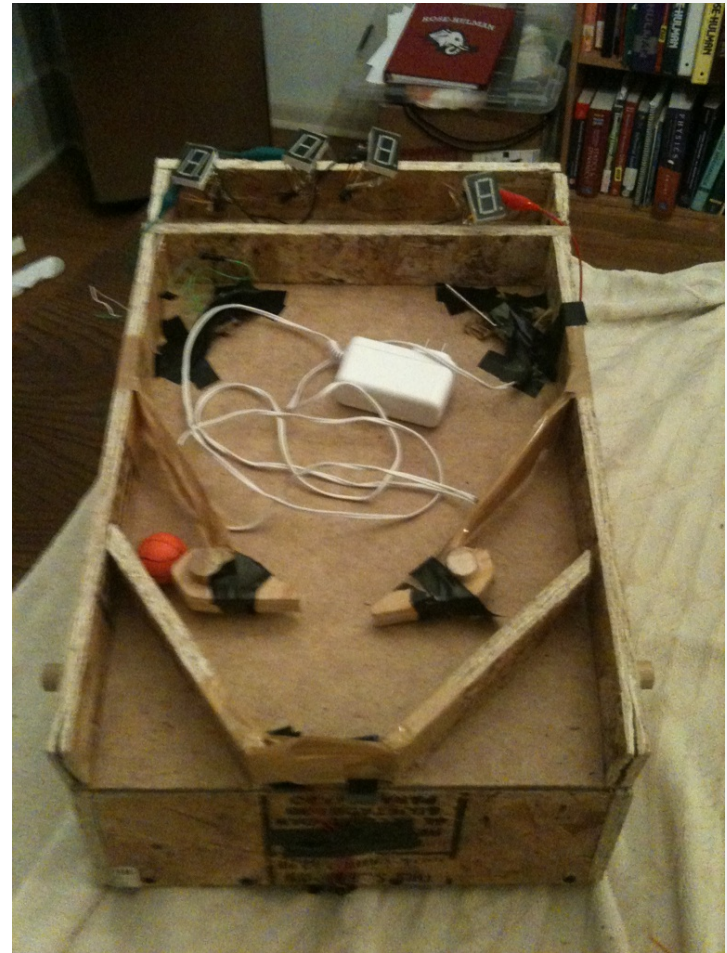
Results

- Worked as designed, we were really happy with our paddle design
- Once a switch was pressed the score increased
- To make a higher score the ones digit was wired to read a constant zero
- Lost ball counter worked, but had an eight second delay to compensate for the lost ball bouncing on the switch and hitting it multiple times
- After the third lost ball, score and lost ball counter all reset to zero



Advice

- Don't put off building and wiring so you have more time for the programming
- Order items with plenty of time in advance so when orders fail there is still time to reorder
- Also provide time to build things so your not using tape as walls and barriers
- Buy wire small enough to fit in bread boards so you don't have to struggle each time you plug a wire in



Most Useless Machine

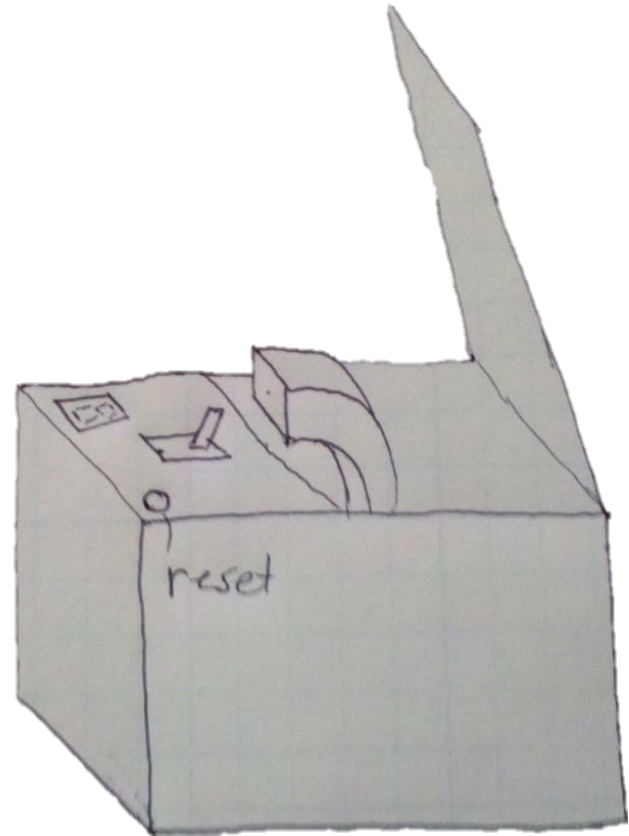
Mechatronics ME430

Andrew Ulrey

Michael Ulrey

Project Objective

- Create a machine that turns itself 'off' after activating the switch
- To further complicate the machine we added seven segment displays to count the number of times it has completed a cycle
- The counter will have a reset switch

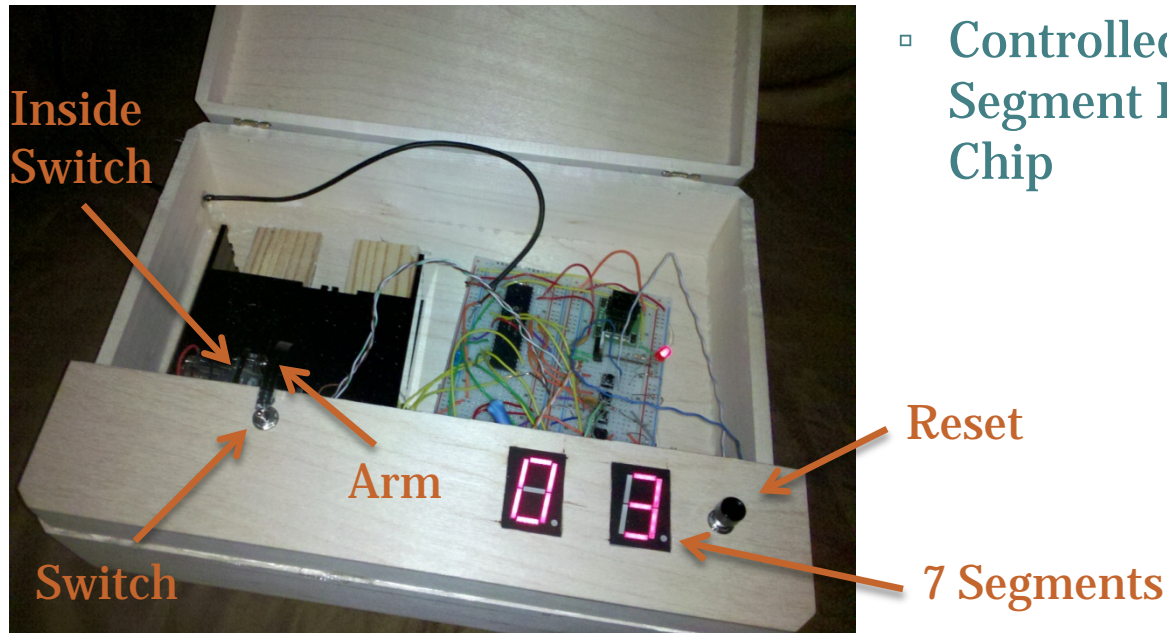


Inputs

- A switch that is first activated by a user then the arm will flip the switch back
- Counter reset button that will set the counter back to zero
- Switch inside to turn off the motor

Outputs

- Geared motor attached to the moving arm
 - Attached to an H-bridge to allow bidirectional control
- The seven segment's displaying the number of times the machine has ran the cycle
 - Controlled by BCD to 7 Segment Decoder/Driver Chip



PIC Setup

- Interrupts on change for the following ports:
 - RB0 is used for when the switch is pushed
 - Will then run the motor until the signal goes low again from pushing the switch back
 - RB1 for the switch inside the box
 - Turns off the motor completely and counts the 7 segment displays
 - RB2 for when the reset button is pushed
 - Resets the 7 segment to zero

Results

- The device was fully functional as submitted in the project proposal
- Initially had difficulty from the motor creating surges and as a result caused the PIC to erroneously believe the inside switch was being pressed

Advice

- When working with motors always ensure to use snubber diodes even if it doesn't seem necessary because it can cause power surges throughout entire circuit
- Pre-drawing a circuit diagram early on will save time in the long run

MUSIC BOX

Sabre Marie Evans

ME430- WINTER 2010

DR. OLSON

Project Objectives

- The project was originally designed to be a musical instrument however due to human error and limitations on the project now plays various song selections at the touch of a button.

Inputs and how they were sensed

- The inputs consisted of:
 - push start to enable users selection preference

Outputs and how they were made

- The output were a visual display on LCD of song selection as well as songs played through a peizo buzzer.

How did I use the PIC as a controller

- The PIC was connected to the following:
 - Voltage regulator
 - A push button used to adjust the song selection
 - LCD
 - Piezzo buzzer
- ADC channels were not needed.
- Program used PWM to control pitch and volume. Timer 0 and where used to indicate if a new selection was made. Interrupts were also used.
- Two main if statements were then outlined one in the main function that feed into the other which was an interrupt. If the button was pressed the song selection was increase until timeo was flagged. depending on song selection index. If the song selection index exceeds the selections available the index returns to 1. Individual functions control the LCD as well as others that list the song notes.

Did it work

- The project did not work. Wiring for the LCD proved to be a little more difficult than expected and in the final stages needed pins on the PIC were burnt out. However modifications were made to allow for the code to be ran on the board.

Advice

- Begin construction early even if you are unsure of how to code for your project or what will be needed on your breadboards.
- Build in plans for it things fail have extra parts incase you blow some out.
- Determine what you have to have the project do and what you would like it to do (i.e. a simple project and if time allows additional bells and whistles.)

Andrew Oakley
Kreigh Williams
ME-430

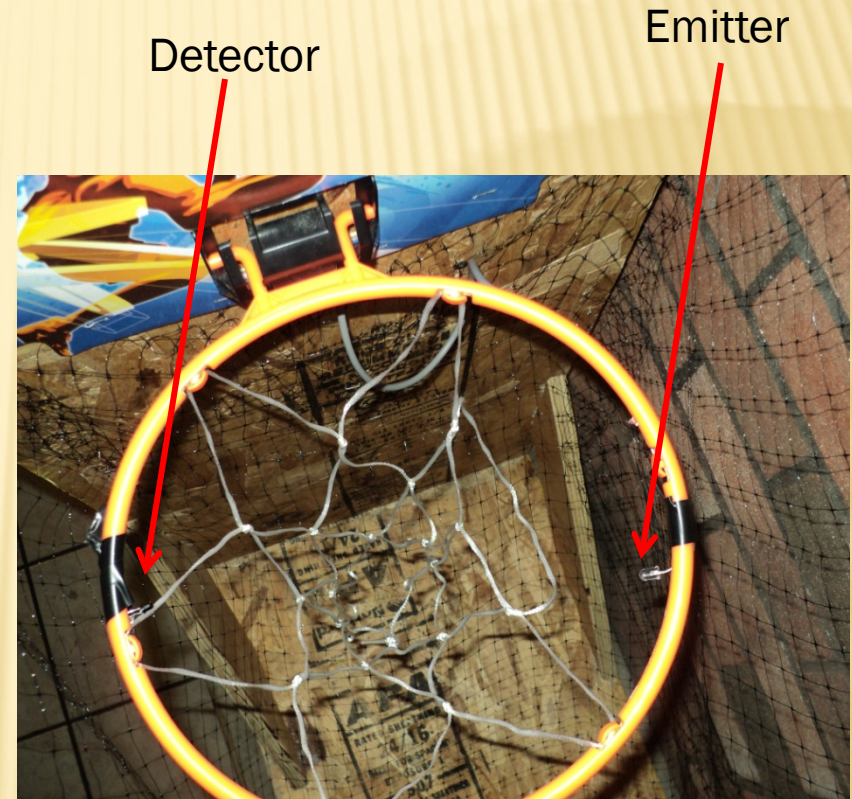
ARCADE BASKETBALL SHOOTOUT

PROJECT OBJECTIVES

- ✖ Make a functioning arcade basketball game
- ✖ Have the clock count from 60 to 0 and then reset back to 60
- ✖ Have the clock begin counting when the first basket is made
- ✖ Count the baskets
- ✖ Have a buzzer sound at the end of 60 seconds

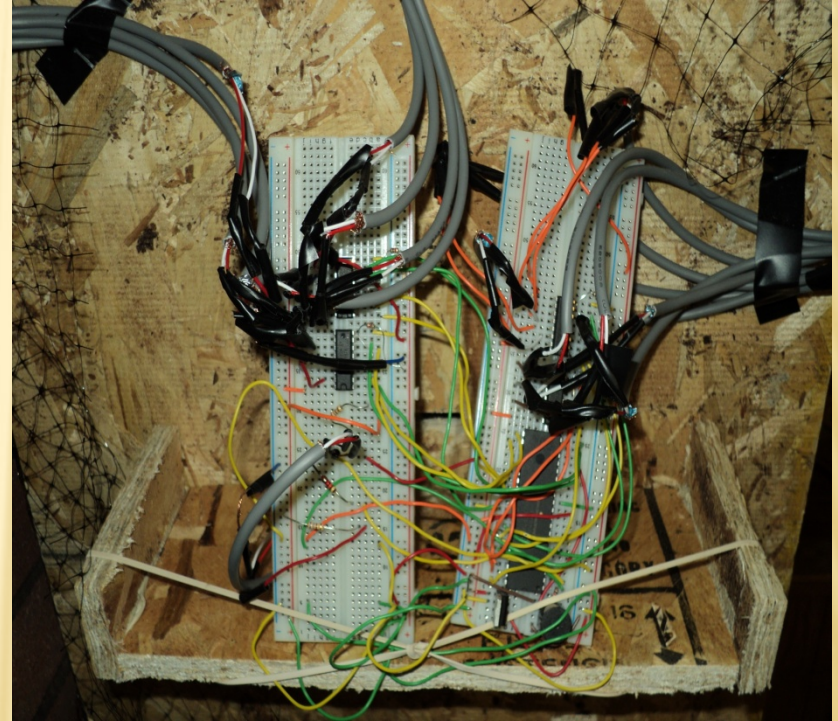
INPUTS

- ✗ The system needed to know when a shot was made
- ✗ An IR emitter and receiver system was used to accomplish this
- ✗ This input would need to start the timer on the first basket, and also change the counter every time a basket is made



PIC COMPONENTS

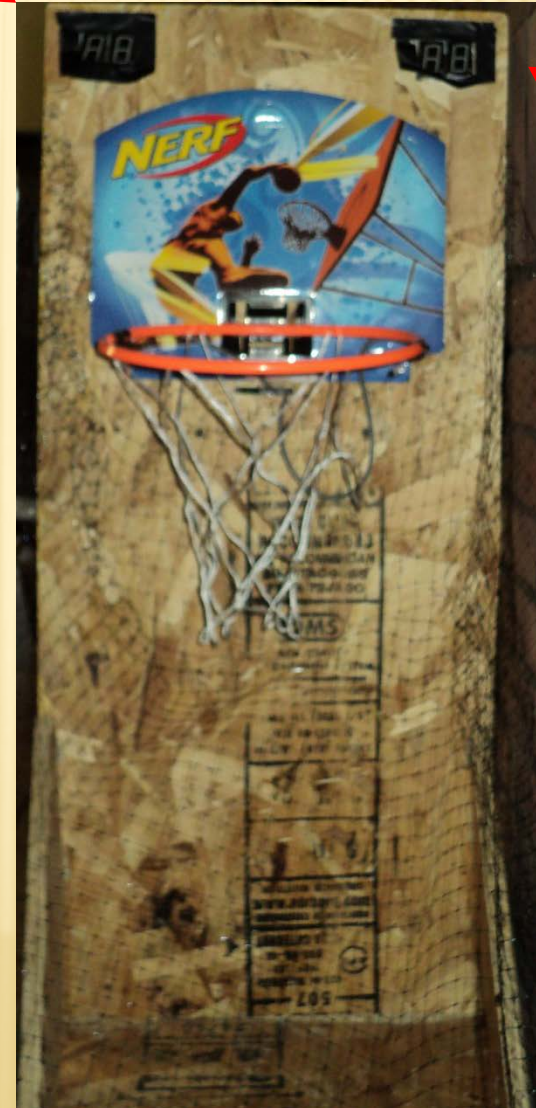
- ✗ Reads the analog signal from the IR receiver
- ✗ When the timer reaches 0, PWM is used to sound a buzzer
- ✗ The timer is counted using a timer interrupt
- ✗ The basket counter uses a delay in the while loop and outputs each value on the LEDs



OUTPUTS

Timer LEDs

- ✗ 2 pairs of LEDs
 - + 1 pair indicates score
 - + 1 pair indicates remaining time
- ✗ Buzzer signaling end of game



Score LEDs

RESULTS

- ✗ All objectives were met
 - + Accurately counted baskets when ball goes through hoop
 - + Time counts down accurately
 - + Buzzer sounds at appropriate time signaling end of game
 - + All LEDs reset to appropriate value after game is over
 - + Clock started counting after first basket was made



ADVICE FOR FUTURE CLASSES

- ✗ Start early
 - + Project will take 2-3X more time than expected
- ✗ Cover all solder connections with electrical tape
 - + Eliminates several variables when debugging
- ✗ Combine electrical and mechanical portions as early as possible
 - + Large amounts of debugging is inevitable during the integration



ME 430: Mechatronics LED Arcade Game

Created By:

Casey O'Neill

Samuel Newman

Objective

- To create reaction-time based game similar to the popular arcade game Cyclone.
- The player would attempt to stop a quick-moving light on a particular “jackpot” light by pressing a button in order to gain points.
- The game would have 3 different difficulty settings.
- The score and other information would be displayed on a LCD screen.



Figure 1: A photo of the arcade game Cyclone, the inspiration for our project.

What inputs did we use?

Figure 2: On-Off slide switch for power



Figure 3: Big Dome Pushbutton for light stopper

Figure 4: On-On-On 3 position slide switch for difficulty setting.



How were these inputs used?

- The big dome pushbutton was set up using the simple switch circuit shown in Figure 5 with its signal connected directly as a digital input to the PIC.
- The difficulty slide switch was three of these circuits.
- The power switch was connected to the power source and the circuit's unregulated voltage. When “off” the switch was an open and would not allow power to the circuit, when “on” it would allow current to pass and power the circuit.

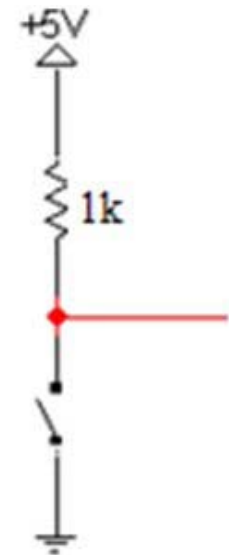


Figure 5: A diagram showing the simple switch circuit used in our project. The red line indicates the signal sent to the PIC.

Outputs



Figure 6: A 40 Character, two line, LCD with LED backlight was used to display score and other messages. It was connected directed the PIC's digital outputs.



Figure 7: 5mm red LED used for the “jackpot” light.



Figure 8: 19 3mm red LEDs used for non-jackpot lights, power light and difficulty lights.

PIC

- Interrupts were used to detect if the big red button had been pressed. After the interrupt is triggered, the PIC would stop the movement of the lights, check which light it had stopped on and assign points accordingly.
- The LEDS were controlled by digital outputs of the PIC connected to Darlington transistor arrays.
- Delays were used to control the movement of the LEDS. The difficulty switch controlled the duration of the delays and changed the rate at which the lights moved.
- The score and other information were sent directly from the PIC to the LCD screen and displayed with the help of "LCD Module.c" and "LCD Module.h".

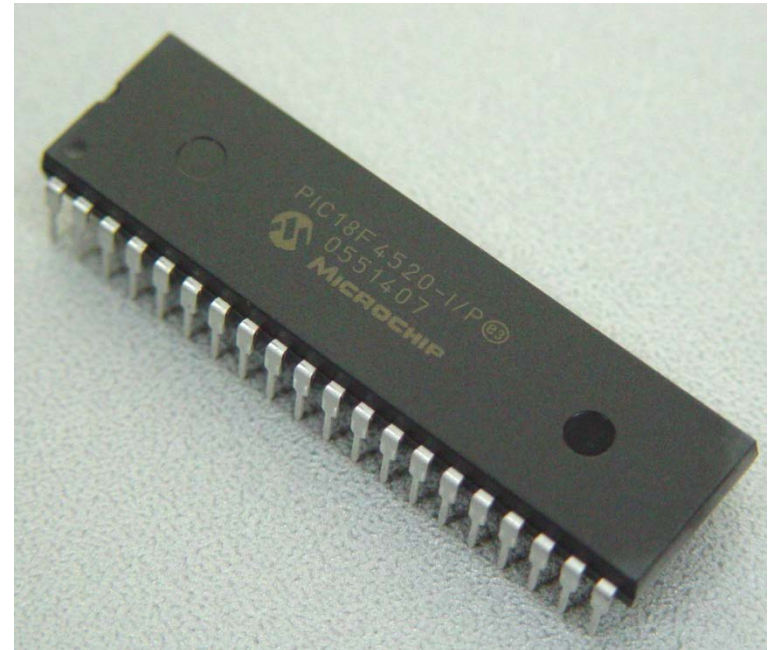
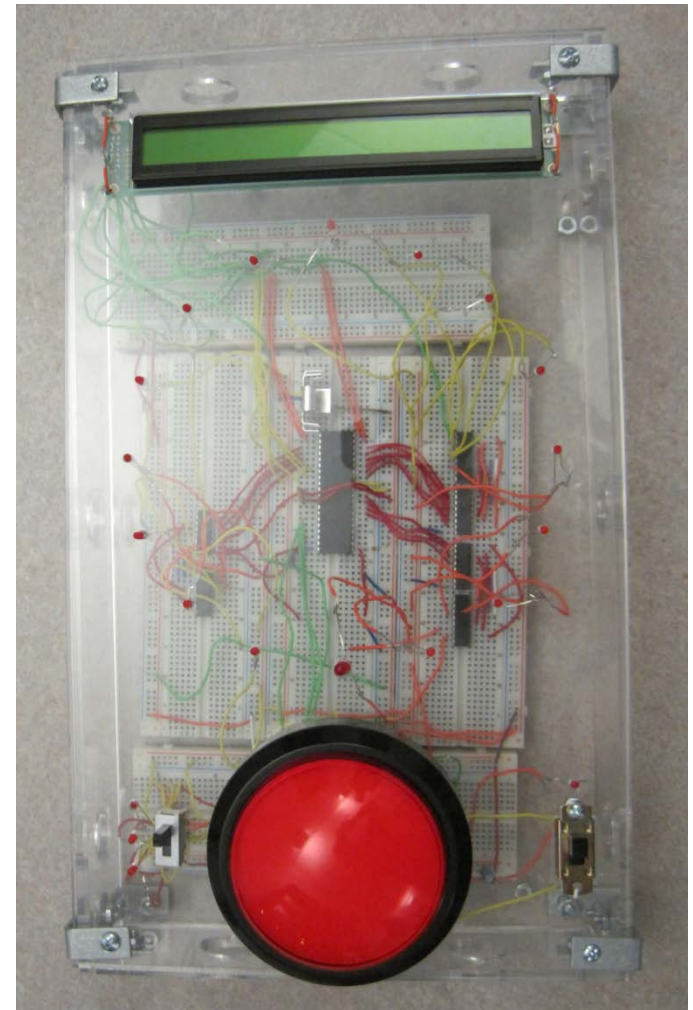


Figure 9: A photo of the PIC18F4520 microcontroller used for our project

Results

- The project worked as planned except for minor bug with the scoring; it would sometimes register the score incorrectly if the score was negative.
- The slide switches were slightly buggy when in between states; however this had only a minor impact on the functioning of the device and did not compromise any of our project objectives.

Figure 10: Photo of our finished project (slightly disassembled). The housing was produced from a .220 acrylic sheet and cut using the laser cutter at Rose-Hulman.



Advice

- If you are using an enclosure for your circuit make sure you leave room to access it if you think you need to. One of the most difficult parts of the project was assembling it. During assembly wires came loose and it was difficult to access the circuit to fix it.
- Start the project earlier in the quarter so you have time to debug unsuspected problems. If we had started the project earlier we could have identified and fixed the scoring bug easily. Also, we could have implemented all or most of our extras.
- Start simple and add features as you go. Our project goals were simple; it would have been very easy for us to overcomplicate the project by adding lots of unnecessary features. We started simple and ended up not having time to add many extra features to our project.
- Before building the circuit, make sure you have everything documented. We were able to avoid wasting a lot time by deciding how all the inputs and outputs were connected in the circuit before we built it. Being organized and documenting things helped us a lot throughout the project.



Crazy Car

Meshari Alquraini

Sean Maher



Overall Objective

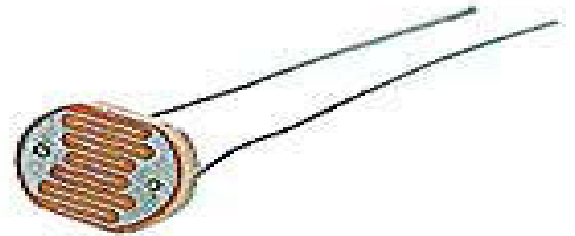
- A car moving forward and lighting when the room is dark.

Parts Used in the project

- 5v voltage regulator
- Breadboard
- Wires
- Capacitor 0.1uF
- Two Darlington
- Four tires and two connecting shaft
- Twelve LEDs
- Battery Pack
- Battery
- Gears
- CDS
- DC Motor
- Resistors (260 ohms, 1000 ohms, and 10000 ohms)

Inputs

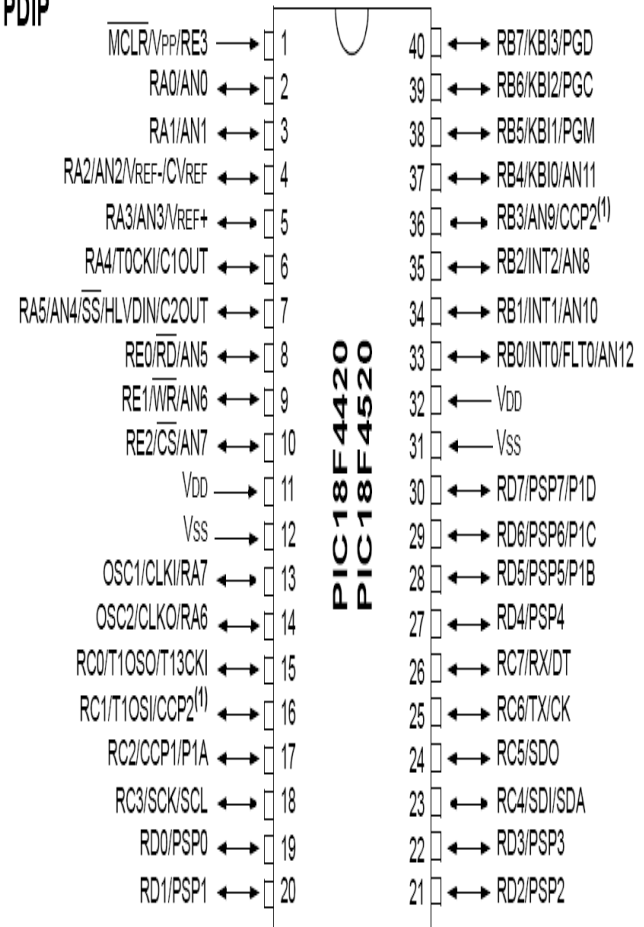
- The CDS is working as switch. It sense the light.
 - If the light is on, it is not going to change anything.
 - If the light is off, it is going to send a signal to the PIC.



Using the PIC as the Controller?

- There is one input, and it is connected to pin2 which is the CDs connection.
- When the PIC receives the signal from the CDS, it sends a current to the output pins.
- There are 14 output pins connected to the PIC.
- The output pins are connected to two Darlington and MOSFT.

40-pin PDIP

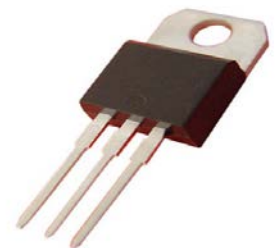
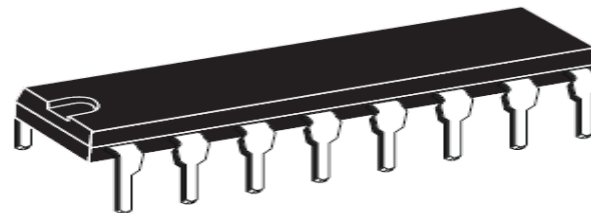


Programming the PIC

```
/** Processor Header Files *****/
#include <p18f4520.h>
/** Define Constants Here *****/
#define lightoff 0
#define lighton 1
/** Local Function Prototypes *****/
// =====
// Configuration Bits
// For details on PIC18F configuration bit settings, see
// PIC18 Configuration Settings in MPLAB-IDE Help
// =====
#pragma config OSC = INTIO67
#pragma config WDT = OFF
#pragma config LVP = OFF
#pragma config BOREN = OFF
#pragma config XINST = OFF
/** Declarations *****/
char recentButtonState = lighton;
/*****
 * Function:      void main(void)
 *****/
#pragma code
void main (void)
{
    int num ;
    ADCON1 = 0x0F; // Make sure the pins are digital
    TRISAbits.TRISA0 = 1; // Makes RA0 an input
    TRISAbits.TRISA6 = 0; // Makes RA6 an output
    TRISAbits.TRISA7 = 0; // Makes RA7 an output
    TRISC = 0x00; // Makes Port C bits 0 & 1 output
    TRISE = 0x00; // Makes Port D bits 0 & 1 output
    while (1)
    {
        if (PORTAbits.RA0 == lighton)
        {
            if (recentButtonState == lightoff)
            {
                PORTAbits.RA6 = 1;
                PORTAbits.RA7 = 1;
                PORTC = 0b11111111;
                PORTD = 0b11111111;
            }
            recentButtonState = lighton;
        }
        else
        {
            recentButtonState = lightoff;
            PORTAbits.RA6 = 0;
            PORTAbits.RA7 = 0;
            PORTC = 0b00000000;
            PORTD = 0b00000000;
        }
    }
}
```

Outputs

- Six LEDs connected to each Darlington.
- The MOSFT is connected to a motor.
- When the Darlington and MOSFT receive the current signal from the PIC, the Darlington amplifies the current.
- Then the Darlington distributes the current to power the LEDs.
- MOSFT transfer power to the motor as well



Final output

- ◉ When the LEDs receive the current, the light on.
- ◉ When the motor receives the current it turns on, and it moves the gears attached to the car tire.
- ◉ Therefore, the car movement and LEDs are controlled by the CDs.



Results: did it work as planned?

- The CDS worked well
- The LEDs and the motor turned on when the CDS sensed any level of "darkness"
- The motor was unable to transmit enough torque to drive the car

Advice to future “generations” (classes)

- Start Early
- Do not be too ambitious
- Coloring your wires makes it easier to debug.
- Choose a motor that has high torque, do not worry about the speed, slow speed is still moving
- Choose tires that are significantly larger than the gears being used to avoid ground contact by the gears

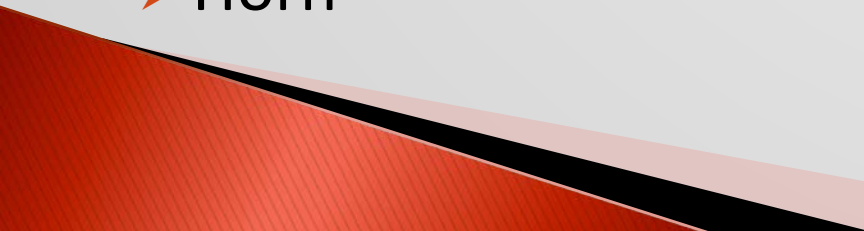
Bike Computer

Petras Swissler

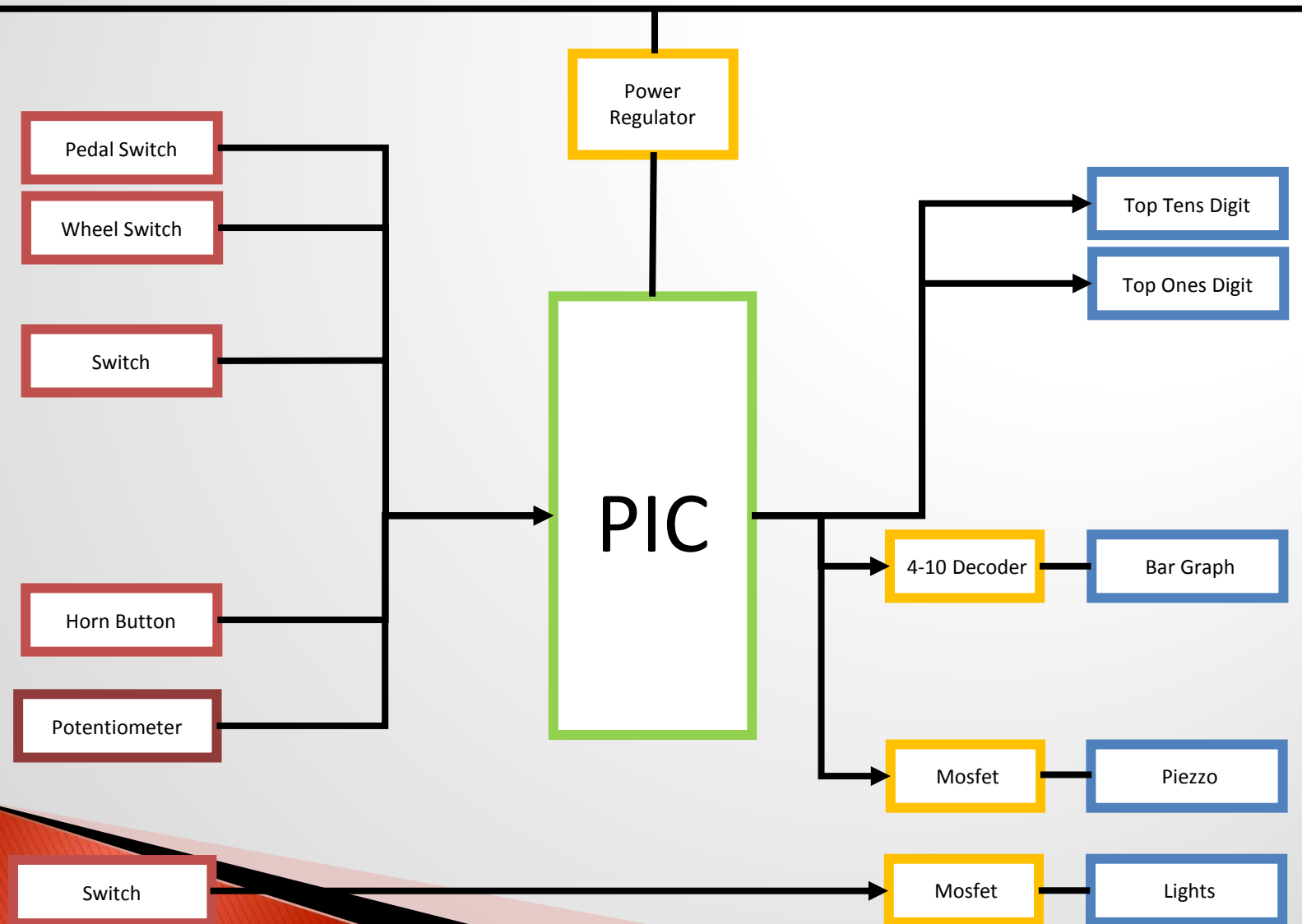
Steven Keltner

Objectives

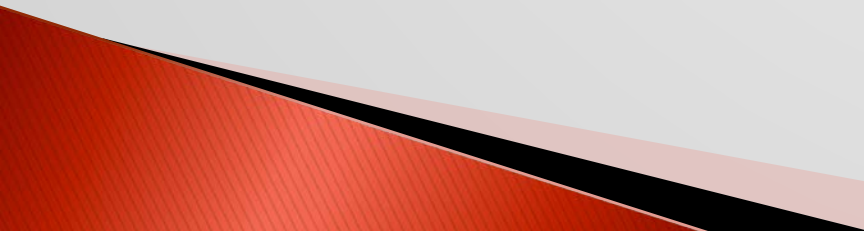
Our overall objective was to make a bike computer which would allow a bike to have many of the features you have in your car. In the end, we decided that we would create a bike computer which would include the following features:

- Speedometer
 - Odometer (With ability to set wheel size)
 - Cadence Meter (How fast you are pedaling)
 - Light control
 - Horn
- 

Top Level System Diagram



Inputs

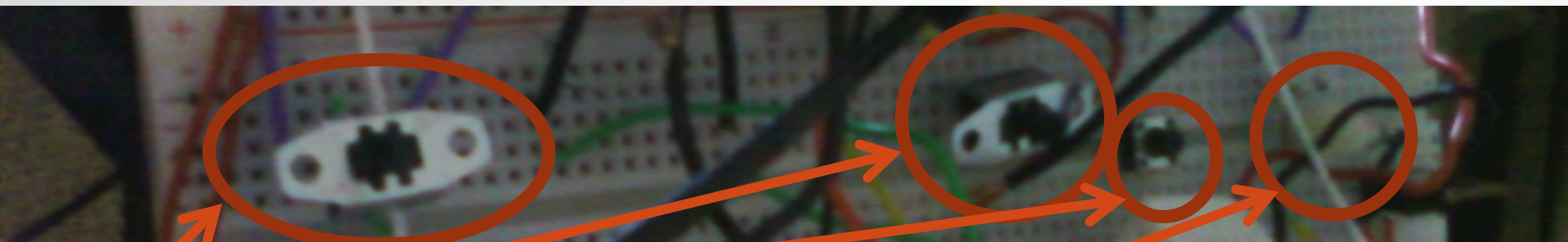
- ▶ Magnetic switches were used to sense when the pedal or wheel made one full revolution
 - ▶ Two switches were used. One was used to enter wheel size change mode, and the other was directly connected to the lights
 - ▶ A button was used to activate the horn and, when in wheel size change mode, to set the wheel size
 - ▶ A potentiometer was used to select the wheel size
- 



Magnetic Switch for detecting wheel rotation



Magnetic Switch for detecting pedal rotation



Lights Switch, Mode Switch, Horn Button, and Potentiometer for selecting wheel size

Outputs

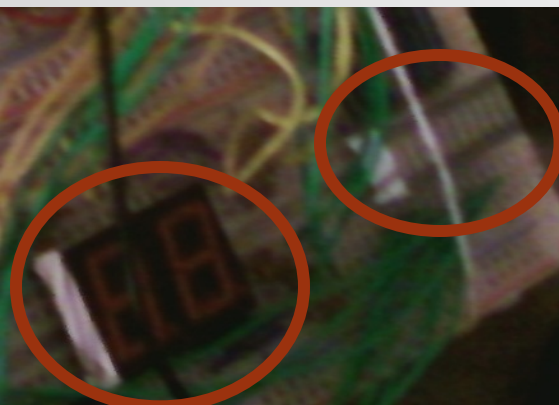
- ▶ A two digit 7-segment display was used to display speed (while riding), distance traveled (while stopped), and wheel size (when mode wheel size is activated)
- ▶ A 10 line bar graph (controlled via a 4-10 decoder), displayed cadence, as well as was used to select wheel size to the nearest 0.1 inch
- ▶ A piezzo buzzer was used for the horn, and was amplified using a MOSFET
- ▶ Super bright white LEDs were used for headlights, and standard red LEDs were used for tail lights. Lights were run off of a MOSFET



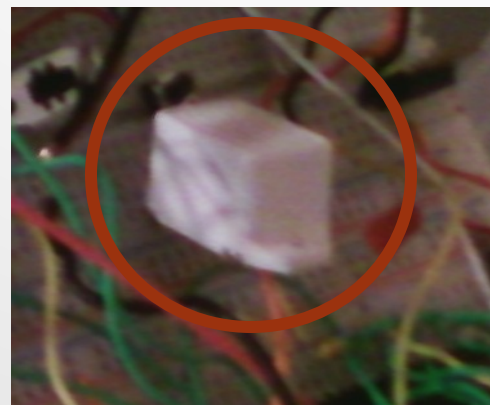
The front headlights are bright enough to illuminate heritage trail at night



The taillights are very visible, even during the day, and increase safety



The 7-seg display shows speed or distance traveled, and the bar graph displays cadence or a decimal value



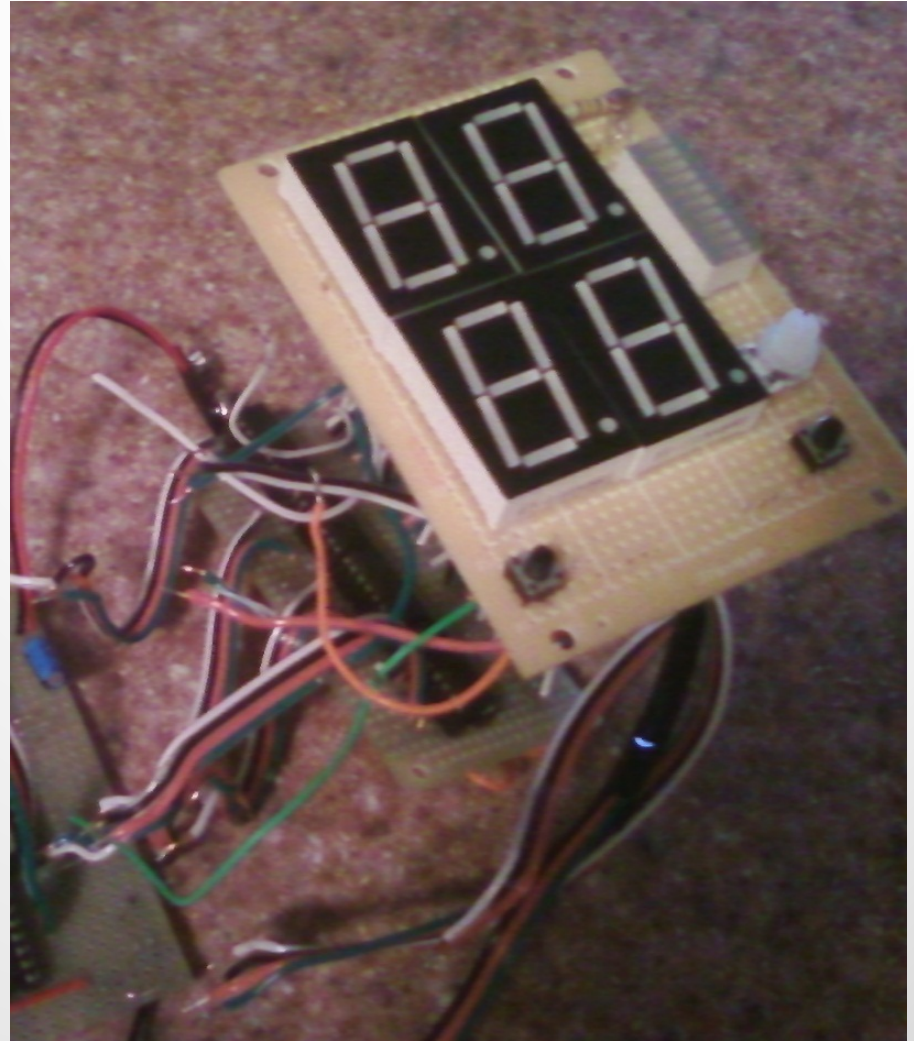
The piezzo buzzer is loud enough to alert pedestrians

Program Overview

- ▶ Interrupts were used to sense cadence and wheel rotations. This was then used to calculate the speed and cadence, and output to the displays. A speed or cadence of zero was determined when a timer timed out
- ▶ The while loop constantly checked the horn button. This was not on an interrupt, as the button was also used for other functions
- ▶ Using an ADC channel, the potentiometer was used to obtain a value of 0-1023. This was then used to select a wheel size from 12.0" to 36.0"
- ▶ A switch was used to switch between normal riding mode and wheel size select mode.

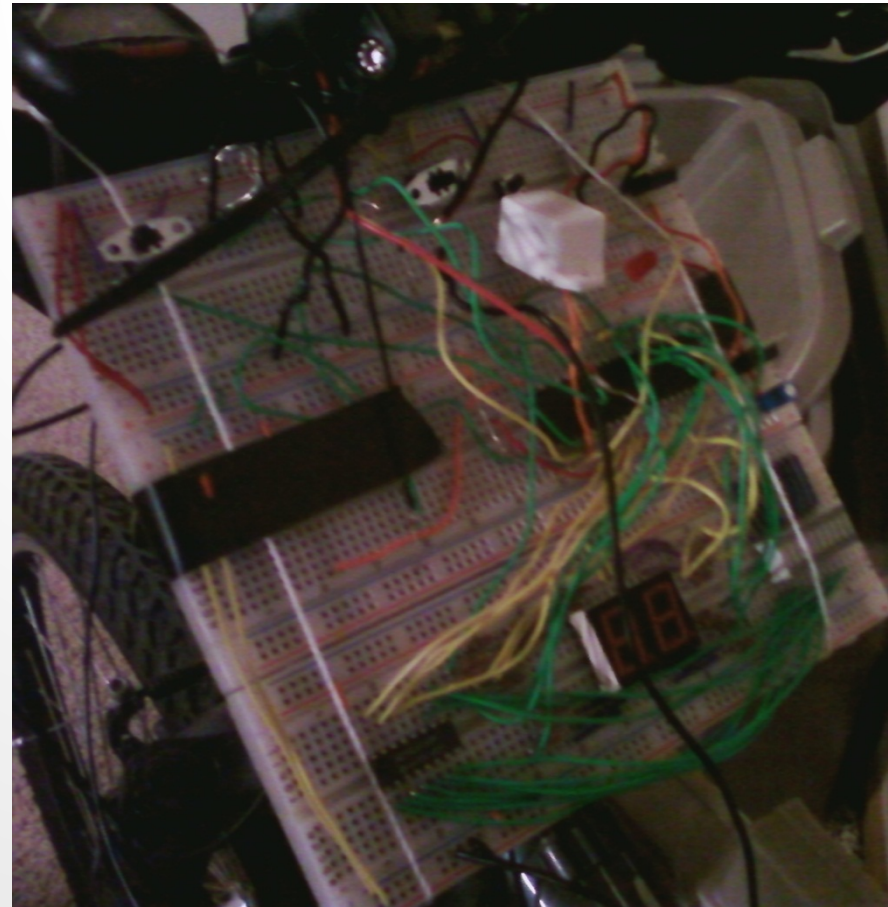
Planned Hardware

- ▶ We had planned to have several extra features (resettable trip odometer, darkness-activated lights)
- ▶ Unfortunately, when we soldered everything together, we had an undiscoverable short and had to abandon the hardware

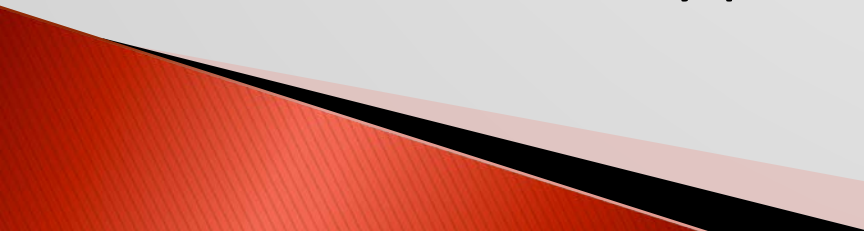


Final Hardware

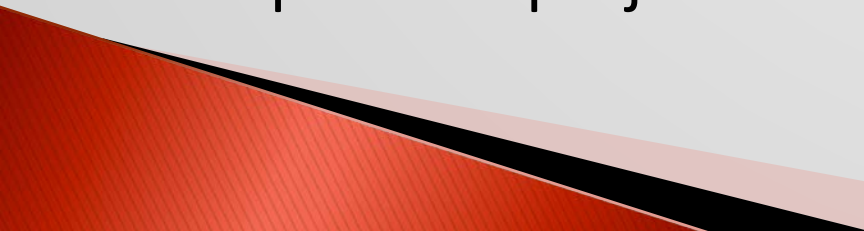
- ▶ Stripping non-necessary features, we used scrap parts we had in our possession to rebuild the hardware.
- ▶ Code had to be modified to be compatible with the final hardware



Conclusion

- ▶ Before soldering everything together, all planned features were fully functional
 - ▶ After having to abandon the planned design, all implemented features worked during the demonstration, with the exception of the cadence meter.
 - ▶ The issue with the cadence meter was later found to be a scaling issue (It was trying to display a cadence greater than the 10 segment bar graph would allow)
 - ▶ In all, we are very proud of the project's performance
- 

Advice to future

- ▶ Whenever soldering, always leave more time than you expect it will take.
 - ▶ Always have excess identical parts, especially when soldering
 - ▶ When soldering, make sure to use sockets
 - ▶ Never use parts from a functioning design to build a new piece of hardware
 - ▶ If you have a large number of displays, consider an LCD, even if you are unfamiliar with them as you start to plan the project
- 

TEMPERATURE CONTROLLED COFFEE MACHINE

Luke Mehringer
Daniel Shepard

Project Objective

- ④ Create a Coffee Maker that brews coffee to a predetermined temperature
- ④ Create a clock to brew coffee at a predetermined time
- ④ Easily change time and set temperature and clock
- ④ Display time and temperature readouts on an LCD screen

System Inputs

- ⦿ Temperature
 - Temperature read off of thermistor, which is a temperature sensitive resistor
- ⦿ Desired Temperature
 - User inputs desired water temperature using buttons
- ⦿ Safety button to prevent brewing without cup
- ⦿ Clock Time and Brew Time
 - User inputs desired brewing time using buttons

PIC Usage

- ⦿ Timer interrupts drove clock function and LCD updates
- ⦿ ADC input to read thermistor voltage, from which we derived the water temperature
- ⦿ Digital Inputs for buttons
- ⦿ Digital Outputs for driving LCD and activating heater relay

System Outputs

- ⦿ Time and Temp readings on LCD screen
 - LCD displays current time and temperature as well as desired brewing time and temperature
- ⦿ Signal to trigger heater relay
 - Heater hooked up to high output DC power supply
- ⦿ Signal to H-bridge to govern motor for opening and closing valve

Results

Challenges: Unreliable valve caused severe issues and made testing and running nearly impossible

Solution: Use a proper solenoid valve

End Result: Excluding the valve, all components of the system worked as intended. With a working valve, system would've functioned correctly

Advice

- ⦿ Do not attempt to design your own valve
 - Solenoid valves rated for high temperature water are expensive and difficult to find
 - Get one anyway, it's far less trouble in the end
- ⦿ Begin working on your project early

Wheel Of Fame

Erik Hoeg

Chad Welty

Project objectives

- Create a Chutes and Ladders type game
- Build a structure capable of holding the mechanical components
- Create a new and unique way for player to interact with a traditional board game

Inputs and Sensors

- Inputs
 - 4 pushbuttons
 - One to reset game
 - One to spin the wheel
 - One to select number of players
 - One enter button to start game
 - IR sensor to read color difference on the wheel(this determines the movement)
 - Potentiometer to control thresh hold of IR sensor

PIC Control

- Kept track of player location
- Determined order of playing
- Told what players needed to do
- Informed if player won
- The use of interrupts, timers, and PWM

Outputs and creation

- LCD display
 - Was used to tell players what to do
 - Informed on player location on “game board”
 - Told player if he/she won
- Motor
 - PWM was used through a MOSFET to control the rate of speed of the motor as it slowed down after the button input was released
 - Used a snubber diode and decoupling capacitor to remove inductive kick

Results

- Project worked as planned
- We added an extra hardware on/off switch
- Added a software addition mode to view analog sensor values on the LCD
 - Allowed easier adjustment of IR sensor threshold values

Advice

- Stick to the recommended timeline
 - It is very easy to get behind and fall into a deep hole
- Test all components as they are received to make sure they perform as desired
- Write small portions of code before assembling the entire project