

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Mechatronics ME430

“The Top-Ten Trashcan”

Andrew Amrein
Michael Petty

Project Objective - Encourage Recycling

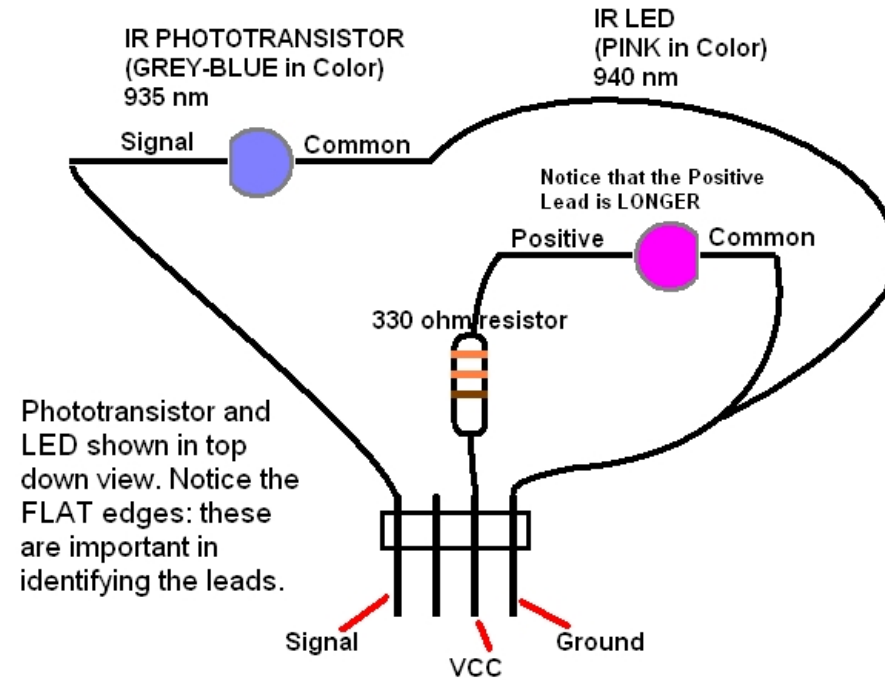
- ▶ We wanted to make recycling fun to encourage more people to recycle
- ▶ We also wanted to be able to track how much recycling was done
- ▶ Our device allows people to set recycling goals and to track their progress



What are the inputs?

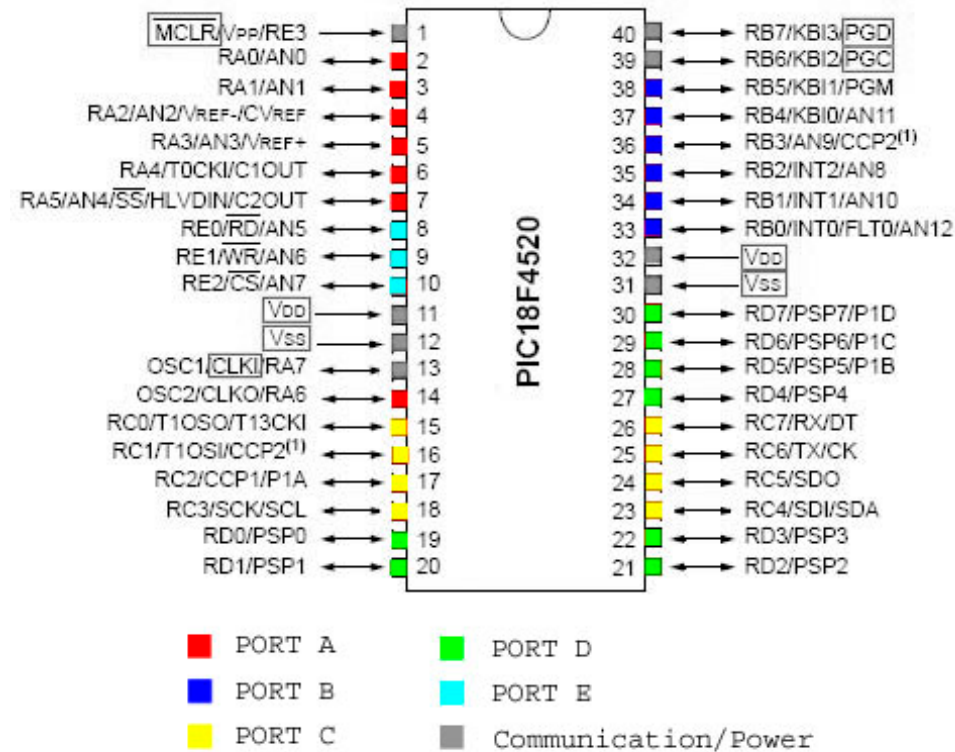
- ▶ There are two inputs - an IR sensor and a button
- ▶ The button is a digital input and the IR sensor is an analog input
- ▶ The button press acts as a reset for the seven segment displays
- ▶ Our program compares the analog values of the IR sensor
 - ▶ When a can is dropped in it breaks the IR beam
 - ▶ When the beam is broken, the value spikes

IR SENSOR ASSEMBLY:



How is the PIC used?

- ▶ We used the PIC in a number of ways
- delays, timers, interrupts, PWM,
and ADC
- ▶ An interrupt was used for the reset
button
- ▶ PWM was used for the buzzer to play
a sound
- ▶ A timer was used for the buzzer
- ▶ A delay was used for the IR sensor
- ▶ Analog to Digital Conversion (ADC)
was used to convert the analog value
from the IR sensor to a digital value
for the PIC



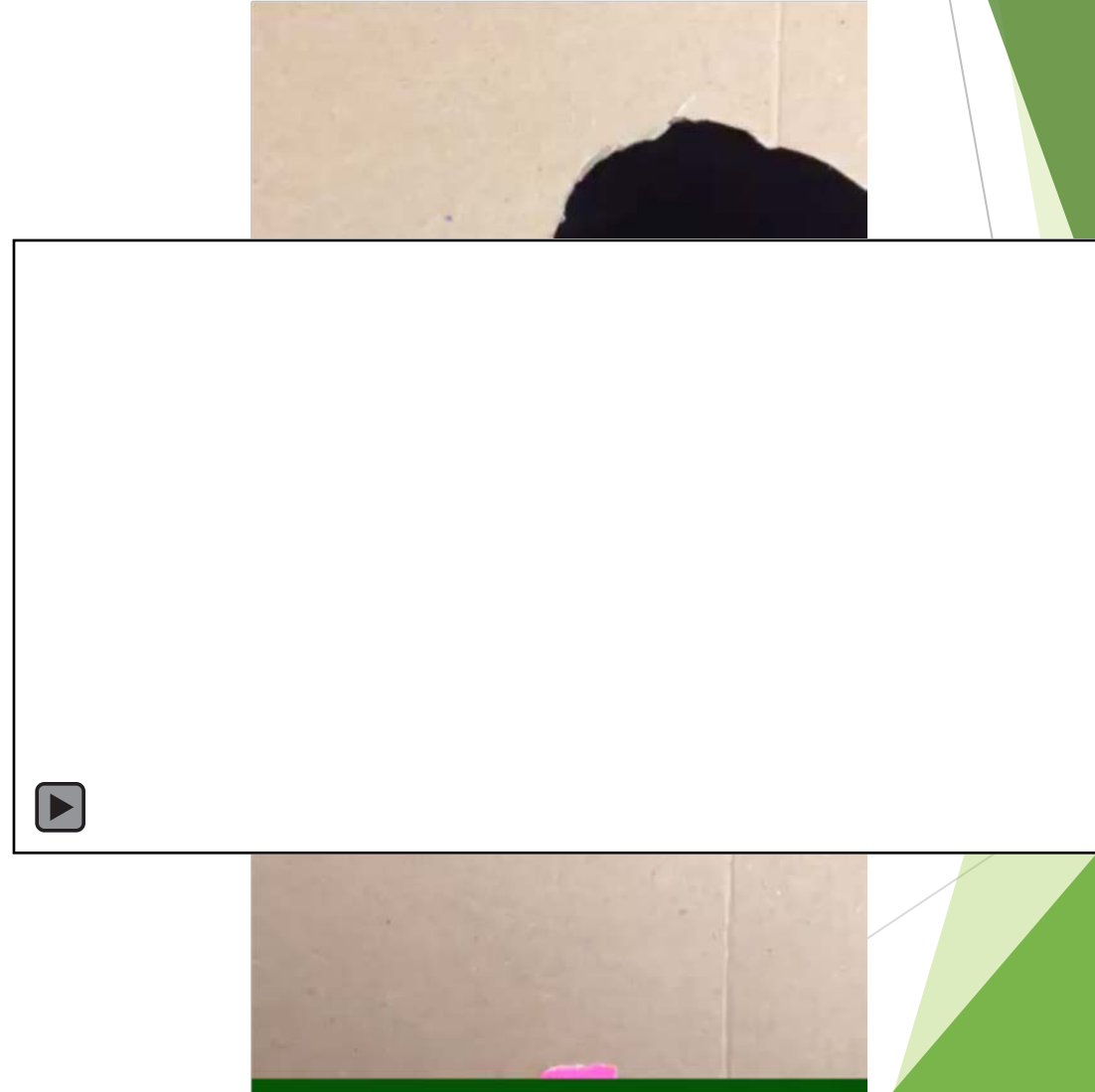
What are the outputs?

- ▶ There are three outputs for our project
- ▶ One of the outputs is a buzzer
 - ▶ The buzzer is triggered when the IR beam is broken
- ▶ Two of the outputs are two seven segment displays
 - ▶ These are created by iterating a counter every time the IR beam is broken



Results – How did it turn out?

- ▶ Our “Top Ten Trashcan” worked perfectly!
- ▶ Dr. Brackin was very impressed with how quickly and how well the IR sensor worked
- ▶ The response of the sound and counter were very quick
- ▶ It worked so well that Michael is actually keeping our project and putting into use at the ATO house
- ▶ To the right is a video



What is some advice for future students?

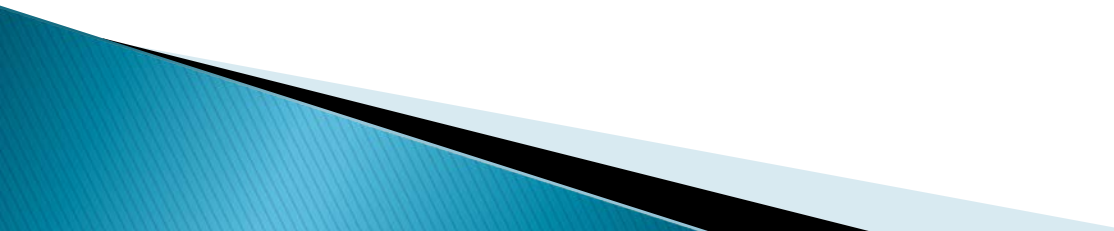
- ▶ GET STARTED EARLY!!!!!!!
- ▶ Pick a project that does not involve too much
- ▶ Pick a project that does not cost a lot of money or require a lot of extra parts
- ▶ Do not pick a project that requires extensive fabrication
- ▶ Talk to people who have taken the class already - they can give you advice on good projects and possibly give you materials for your project



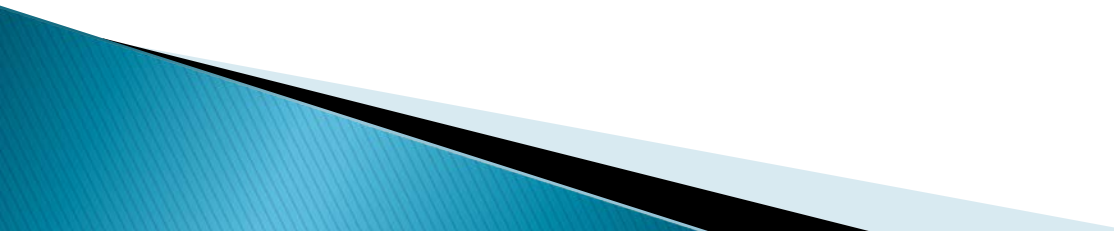
Push Button Candy Machine

Joel Anderson
Caleb Camara

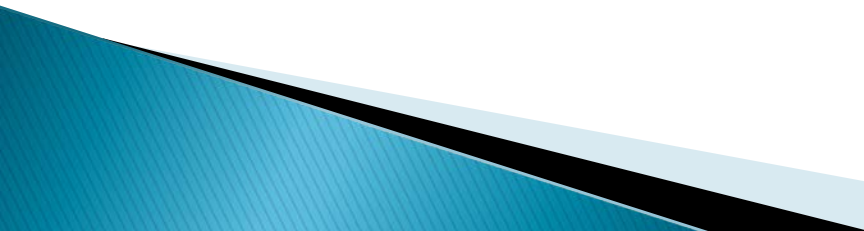
Objective

- ▶ Create a candy machine to dispense small candies
 - ▶ Allow the user to select from 3 different amounts of candy
 - ▶ Allow user to change their mind about how much candy to dispense after having selected an amount
- 

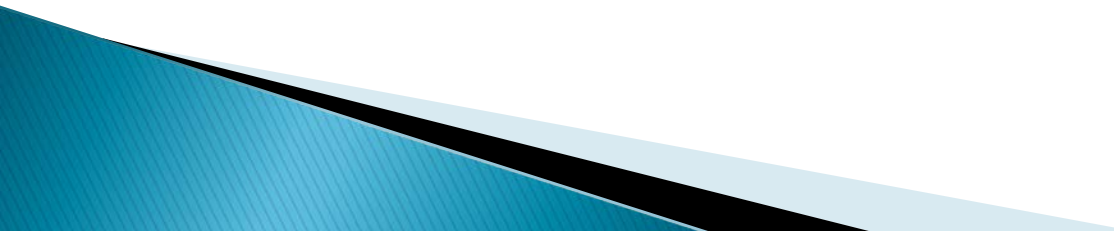
Inputs

- ▶ Three buttons
 - ▶ Set up as simple switches that send a “0” signal to the PIC when pressed
 - ▶ Button would be pushed twice before candy would dispense.
- 

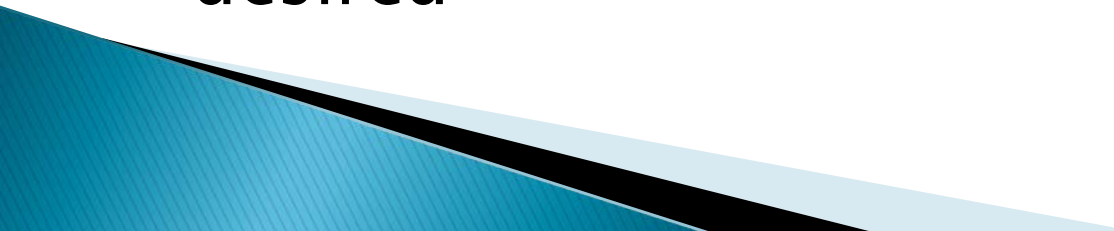
PIC as the controller

- ▶ We used the PIC in a couple different ways
 - Delays
 - Interrupts
 - A function
 - ▶ We used delays with the stepper motor to wait between each step.
 - ▶ We used interrupts for the buttons and implemented a counter variable for the button press
 - ▶ After 2 of the same button press, motor runs
- 

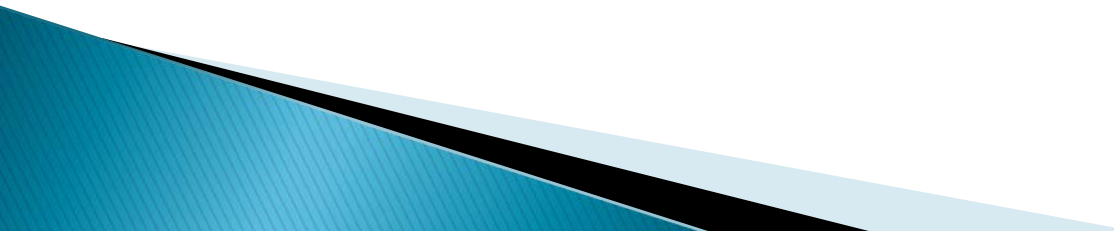
Outputs

- ▶ LED's to indicate which button has been pressed and how much candy will be dispensed
 - ▶ Outputs that control the stepper motor
- 

Results

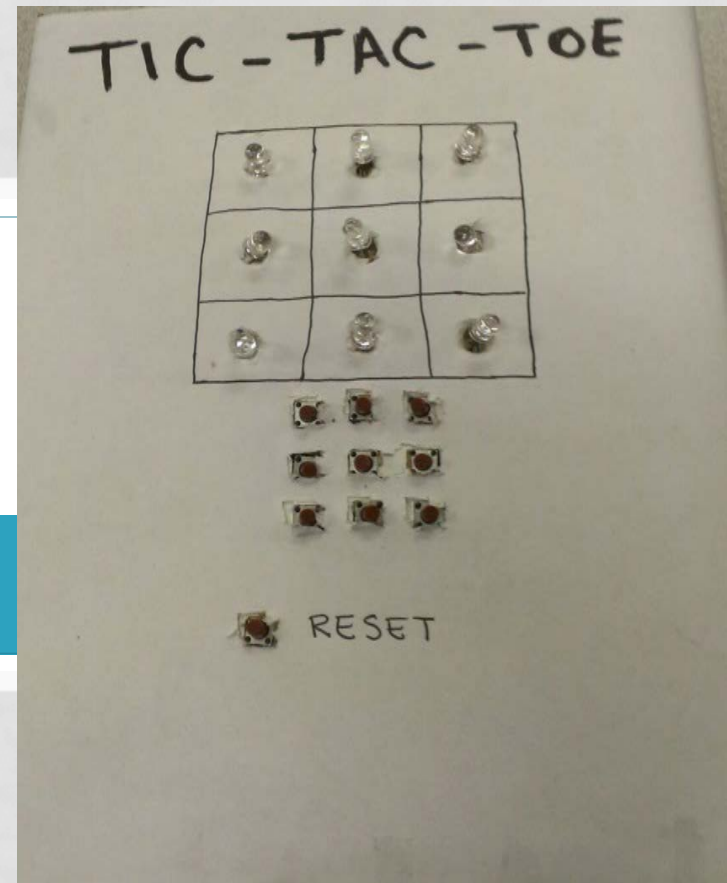
- ▶ We were able to accomplish everything we planned
 - ▶ The mechanism for dispensing candy just rotates farther to dispense more candy depending on the button that is pressed
 - ▶ Buttons had no issues and the LED's lit up as desired
- 

Advice

- ▶ If you want to have time to make your project look pretty, you should start sooner. It's easy to spend extra time at the end to get the project working but there is little motivation to make the project look nice when you're working at the last minute.
 - ▶ Unless you have a lot of extra time for this class, pick a project that you think you'll know how to troubleshoot.
- 

TIC-TAC-TOE

DANIEL CASTLE
CATHERINE HIDLEBAUGH



PROJECT OBJECTIVE

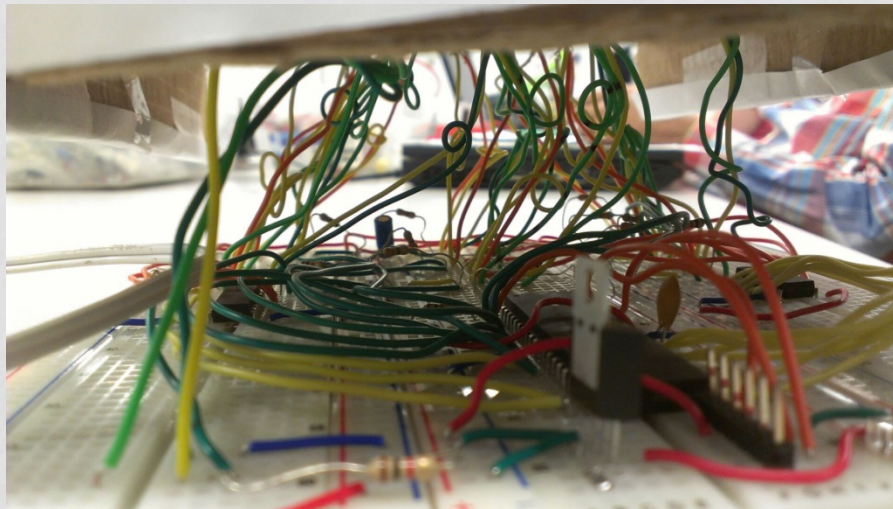
- Create a two player tic-tac-toe game where each player has a specific LED lighting pattern to display once corresponding buttons are pressed. The game can be restarted by hitting the 'reset' button at any time.

PROJECT DETAILS

- Inputs
 - 10 buttons
 - 9 buttons controlled the LEDs, 1 controlled the reset
- How did you use the PIC as the controller?
 - We used the standard IO function of the PIC, as well as the timer interrupt function
 - Timer interrupt controlled the flashing of Player 2's LEDs
- Outputs
 - 9 LEDs
 - When Player 1 presses a button, the LED turns on
 - When Player 2 presses a button, the LED flashes on and off
 - If a button has already been pressed, nothing happens

RESULTS

- Results
 - Worked exactly as planned, though we encountered some issues getting the cosmetic faceplate to fit over everything (too many wires)
 - The buttons were not robust enough for extensive play



FUTURE WORK

- Multicolor LEDs, rather than flashing vs. not flashing
- An indication of when someone has won a game, such as flashing the winning row of LEDs
- Use the buzzer to play a triumphant victory song
- Make the design more compact and more robust

GUIDANCE

- Future advice
 - Start your project **early** – and think through it well before you start it!
 - If using a cover such as a box for lights or buttons, we suggest wiring and soldering everything and THEN pushing the buttons or LEDs through holes in your cover – this way you don't loosen wires or unplug them
 - Take good notes on the videos – a lot of the things you learn from them that aren't used in lab you will use on your project
 - A timer interrupt can be a very useful tool – be creative

Memory Game

Kahan Gandhi and Rachel Vinson

ME 430

Winter 2013-2014

Project Objectives

1. To build a functional memory game using an LCD, multiple LEDs and pushbuttons for the user interface
2. To have multiple levels with increasing difficulty to engage the player

Inputs

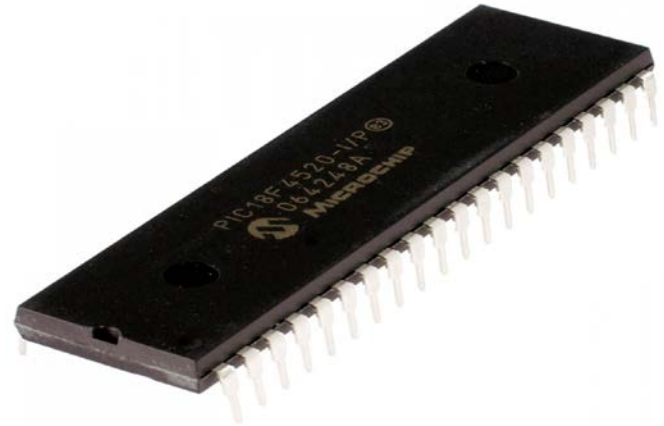
- A pushbutton to start and reset the game
- Four other pushbuttons paired with an LED each as indicated in the picture
- The input was sensed when a button was pressed



PIC as controller

Used finite state machine analogous to the 430 combination lock demonstrated in the “FSMs for PICs” lecture

Used the timer to generate a random number to light the LEDs randomly

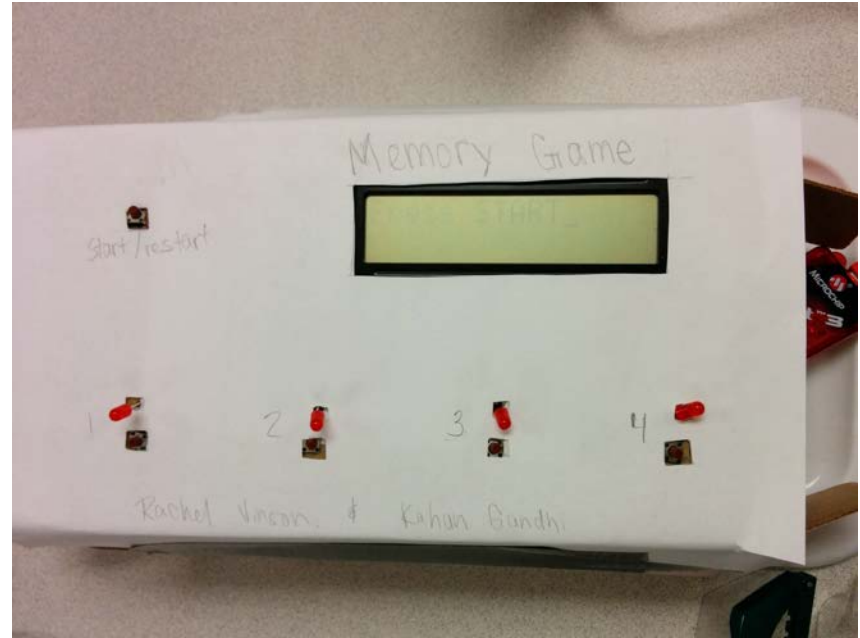


Outputs

- The LCD displays the different levels, incorrect button press and wins as well as instructs to start the game
- The LCD outputs, printed using *XLCDPutRomString* command, are “Press START,” “Level 1” to “Level 10,” “Incorrect! GAME OVER” and “You WON!!!”
- The LEDs flash in a random order and the number of LED flashes increments at each level

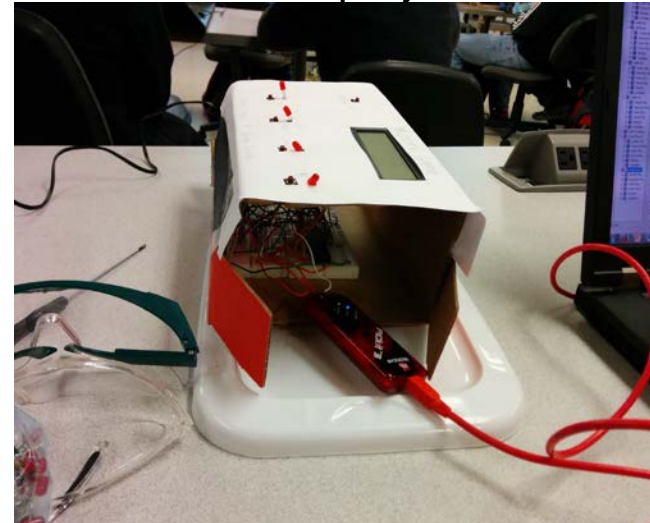
Results

- The LEDs flashed randomly and incremented after each correct button press
- The game resets at all stages
- The LCD display was not very bright and the larger size of the display could have caused that
- The game worked as planned!



Advice

- Practice soldering in class, can save a lot of time if you are good with it
- Check sizes before ordering parts, our LCD turned out to be larger than expected
- Also check for detailed part descriptions and what they include, we did not receive wires with our breadboards and almost ran out of wires at the end of the project
- Try and visualize, plan and color code the wiring before starting on it as it can get very messy



Carnival Target Shooting Game

Yaqoub Khouja

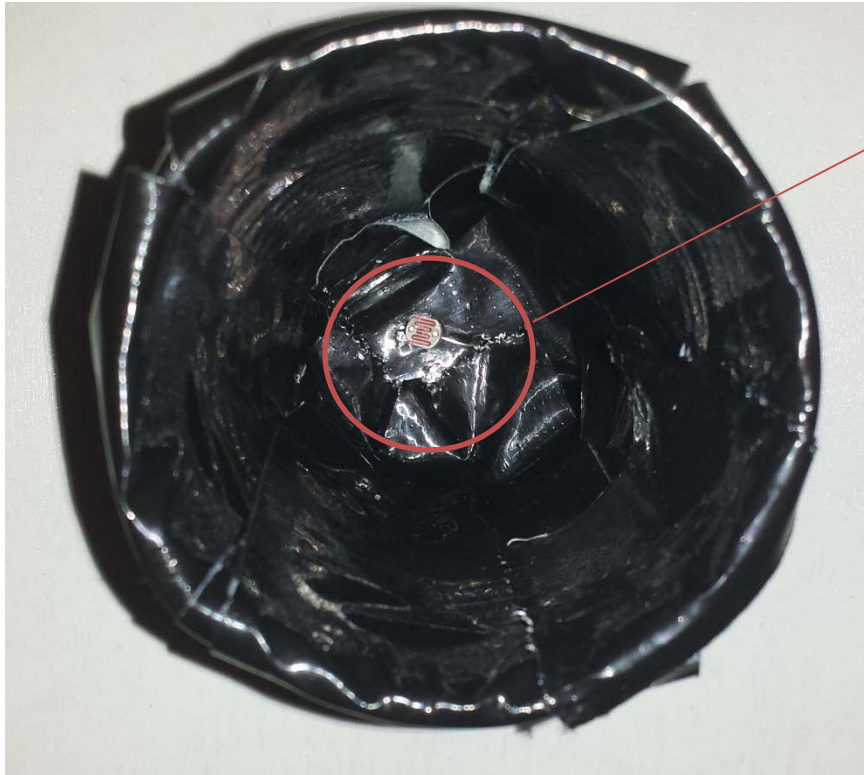
Josh Meeks

Trent Stephens

This project was designed to create a carnival game using a laser to shoot targets in a set amount of time while keeping track of score



The primary input for the game were CdS photocells which sensed light from a laser



CdS Photocell



The PIC was used in a variety of ways, one of which was use PWM for the end of game and scoring buzzer.

```
void BuzzerScore() {  
    OpenPWM1(E_mid);  
    SetDCPWM1(VOLUME);  
    Delay10KTCYx(1);  
    SetDCPWM1(NO_VOLUME);  
}
```

```
void EndBuzzer() {  
    OpenPWM1(C_mid);  
    SetDCPWM1(VOLUME);  
    PORTB = 0;  
    PORTEbits.RE0 = 0;  
    PORTEbits.RE1 = 0;  
    Delay10KTCYx(250);  
    Delay10KTCYx(250);  
    SetDCPWM1(NO_VOLUME);  
}
```

The PIC used interrupts and timers for formatting the scoreboard and game timer while also signaling the end of game buzzer

```
void high_isr(void) {  
    // Add code here for the high  
    if (INTCONbits.TMR0IF) {  
        INTCONbits.TMR0IF = 0;  
  
        i--;  
        PORTE = j;  
        PORTB = i;  
        if (i == -1) {  
            i = 9;  
            j--;  
            PORTE = j;  
            if (j == -1) {  
                PORTE = 0;  
            }  
  
            PORTE = j;  
            PORTB = i;  
  
        }  
    }  
}
```

```
.  
if (k == 30) {  
    EndBuzzer();  
    INTCONbits.TMR0IE = 0;  
}  
}  
k++;  
WriteTimer0(57724);
```

One of the main uses of the pic was reading the photocell and converting to ADC values to be used, these values were used to determine when a target was hit

```
while (RA2value > RA0value) {
    SetChanADC(ADC_CH0);
    ConvertADC();
    while (BusyADC());
    RA0value = ReadADC();

    SetChanADC(ADC_CH2);
    ConvertADC();
    while (BusyADC());
    RA2value = ReadADC();
    if (RA2value < RA0value) {
        Score();
        BuzzerScore();
        PORTCbits.RC1 = 0;
        Delay10KTCYx(50);
    }
}
```

Lastly, the PIC was used to randomize which targets were signaled to be hit during the game

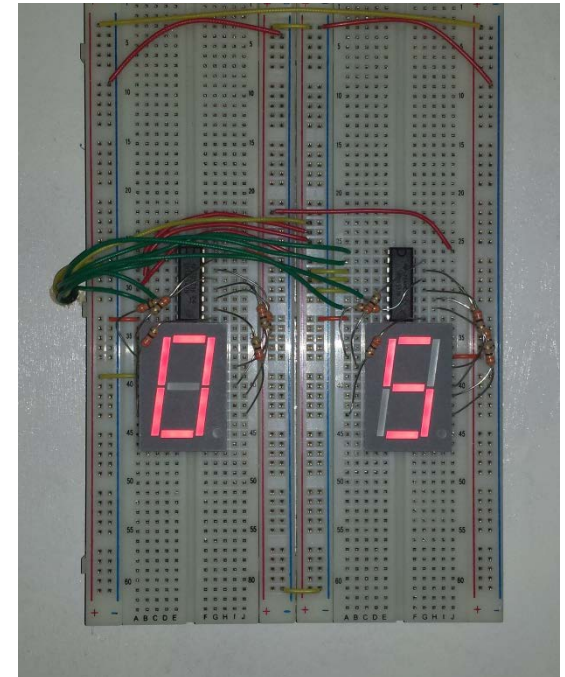
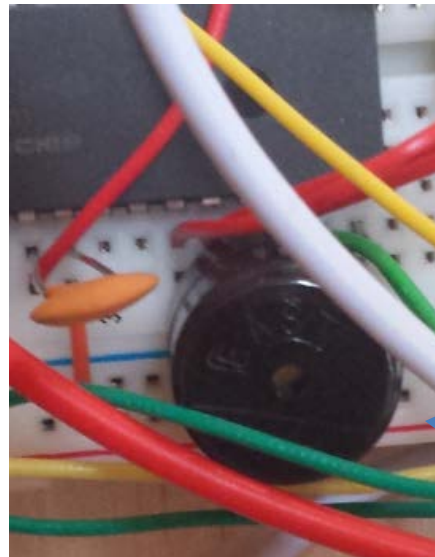
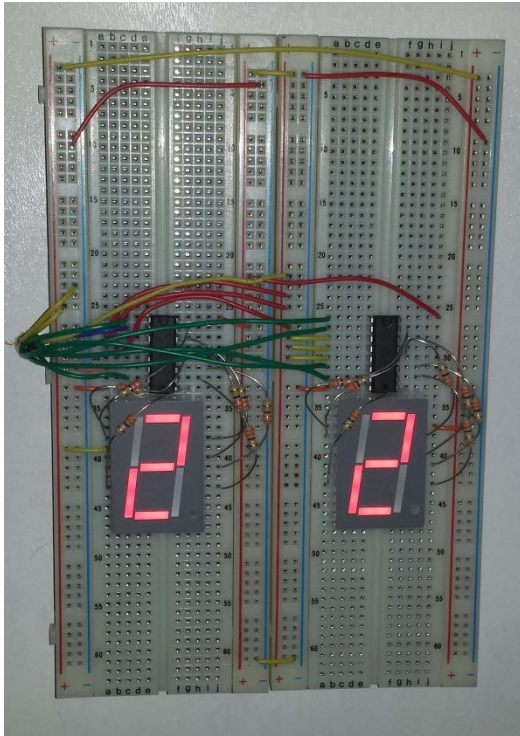
```
randnum = rand();
```

```
if (randnum <= 10922) {  
    PORTCbits.RC0 = 1;
```

```
    } else if ((randnum > 10922) && (randnum < 21845)) {  
        PORTCbits.RC1 = 1;  
        while (RA2value > RA0value) {
```

```
    } else if (randnum >= 21845) {  
        PORTCbits.RC3 = 1;  
        while (RA3value > RA0value) {
```

The game had three outputs, the score, the remaining time, and the sounding buzzer



Buzzer

In the end, the project worked as planned. The photocells turned out to be very sensitive to room light but adjusting them via a potentiometer fixed the issue.



Advice for the future

- Order parts early on in the quarter.
- Separate code into segments to make it easier to debug and fix issues.
- Clearly map out how the hardware will be connected, breadboards become cluttered and confusing quickly.

The background is a light blue gradient with several realistic water droplets of various sizes scattered across the surface. The droplets have highlights and shadows, giving them a three-dimensional appearance.

COOLEST ICE MACHINE EVER

DYLAN LAMBERT

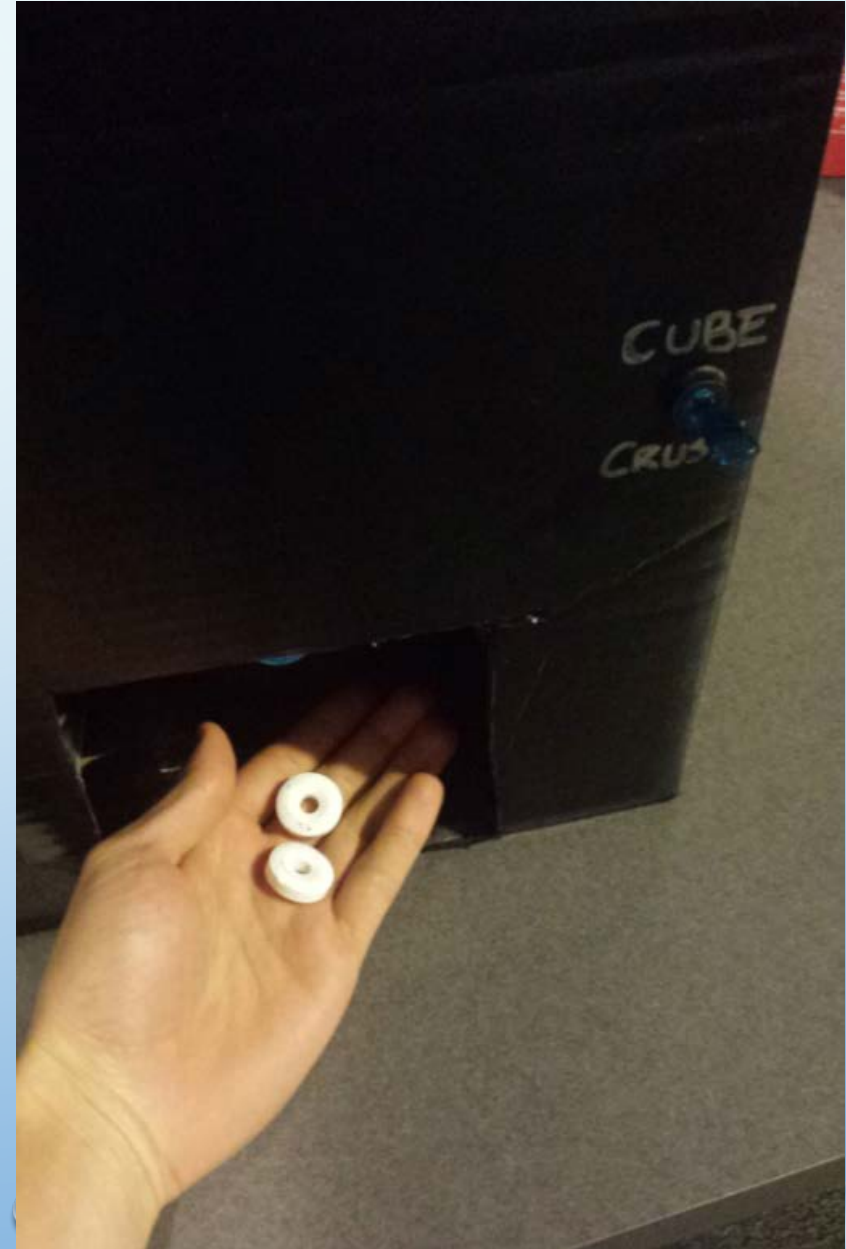
TRAVIS MAY

ME430

2/27/14

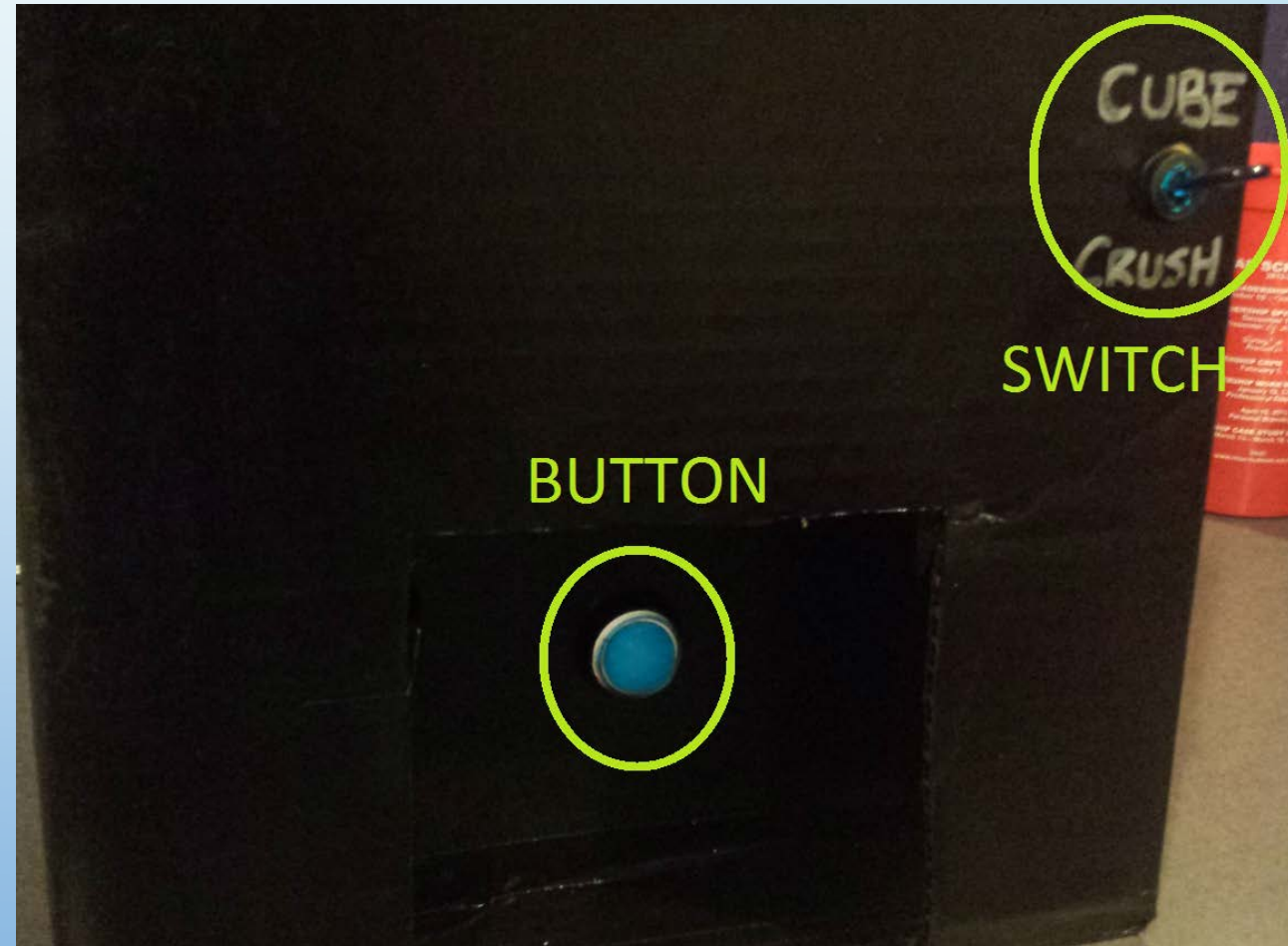
OBJECTIVE

- ALLOW USER TO CHOOSE TO DISPENSE CRUSHED OR CUBED ICE ON DEMAND



INPUTS

- TWO INPUTS
 - ONE SWITCH
 - DETERMINES WHETHER CRUSHED OR CUBED ICE WILL BE DISPENSED
 - ONE BUTTON
 - DISPENSES THE CHOSEN ITEM



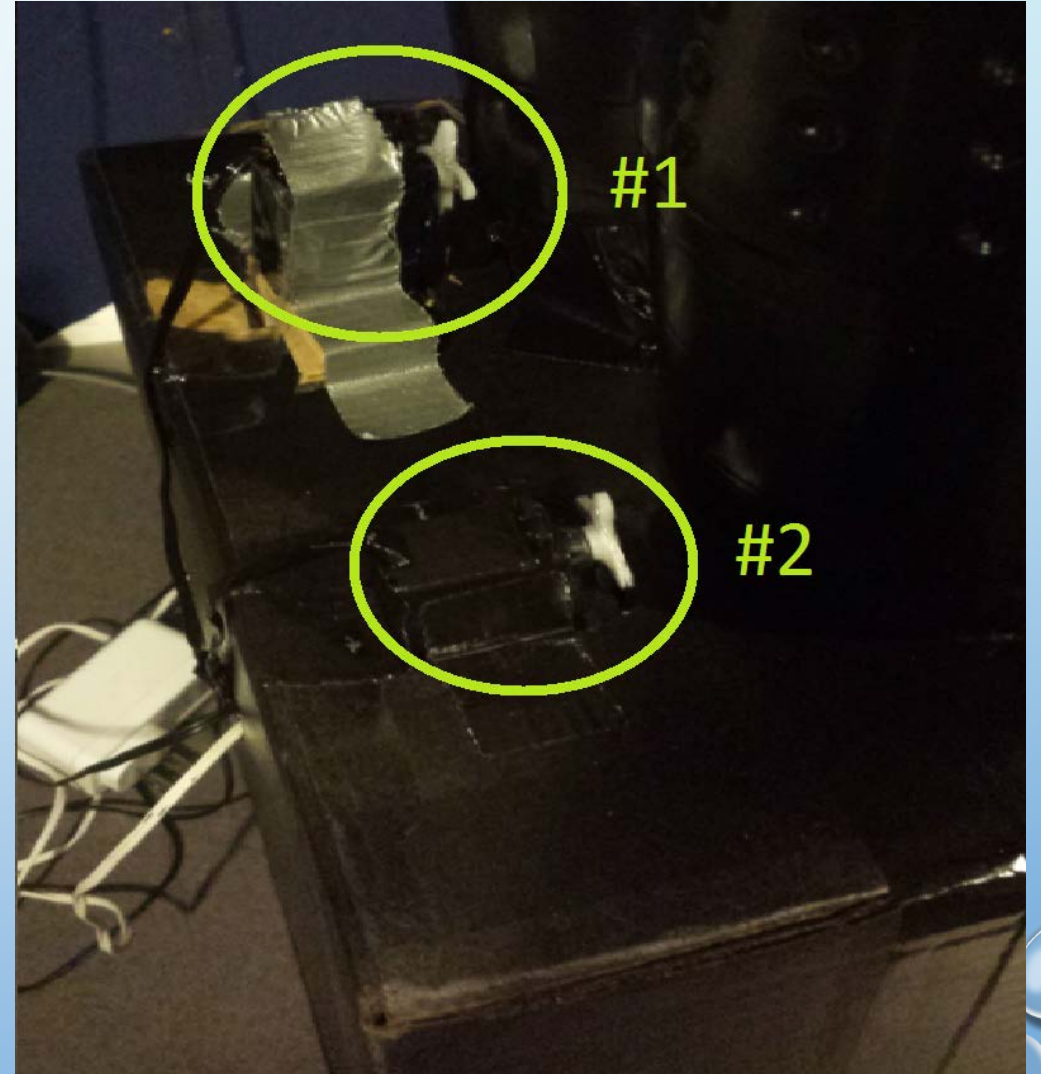
PIC USAGE

- THE SERVOS WERE RUN USING TIMER INTERRUPTS SHOWN TO THE RIGHT
- BUTTON AND SWITCH WERE SIMPLE ON/OFF VARIABLES

```
PORTAbits.RA0 = 1;
PORTAbits.RA1 = 1;
if ((PORTBbits.RB0 == BUTTON_PRESSED) && (PORTBbits.RB1 == BUTTON_PRESSED)) {
    Delay10TCYx(6);
    Delay1TCY();
    Delay1TCY();
    Delay1TCY();
    PORTAbits.RA0 = 0;
    Delay100TCYx(2);
    Delay10TCYx(7);
    PORTAbits.RA1 = 0;
} else if ((PORTBbits.RB0 == BUTTON_PRESSED) && (PORTBbits.RB1 == BUTTON_NOT_PRESSED)) {
    Delay10TCYx(6);
    Delay1TCY();
    Delay1TCY();
    Delay1TCY();
    PORTAbits.RA1 = 0;
    Delay100TCYx(2);
    Delay10TCYx(7);
    PORTAbits.RA0 = 0;
} else {
    Delay100TCYx(3);
    Delay10TCYx(1);
    Delay1TCY();
    Delay1TCY();
    Delay1TCY();
}
PORTAbits.RA0 = 0;
PORTAbits.RA1 = 0;
```

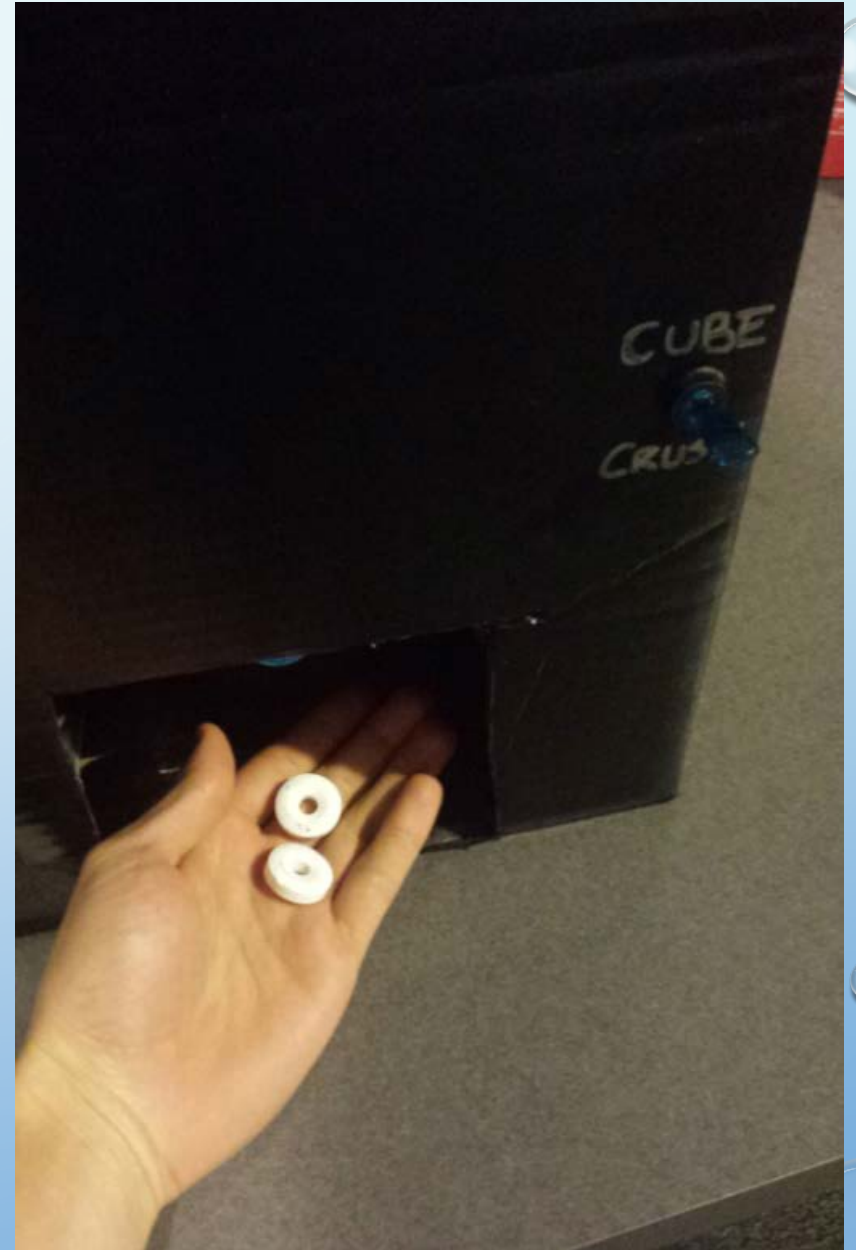
OUTPUTS

- TWO SERVO MOTORS THAT WERE DETERMINED BY THE SWITCH ON WHICH ONE ROTATED
 - TURNED ON WHEN THE BUTTON WAS PRESSED
- SHAKER MOTOR HIDDEN INSIDE PROJECT
 - ON WHEN MACHINE IS TURNED ON



RESULTS

- SUCCESSFUL AT CONTROLLING BOTH OF THE SERVOS ON ONE PIC
- ICE WAS NOT ABLE TO BE USED
 - THIS IS MAINLY DUE TO THE DESTRUCTIVE NATURE OF ICE ON THE BUILD MATERIAL
 - WE WERE ABLE TO SHOW BOTH SIDES DISPENSED AS PLANNED USING A SUBSTITUTE FOR ICE



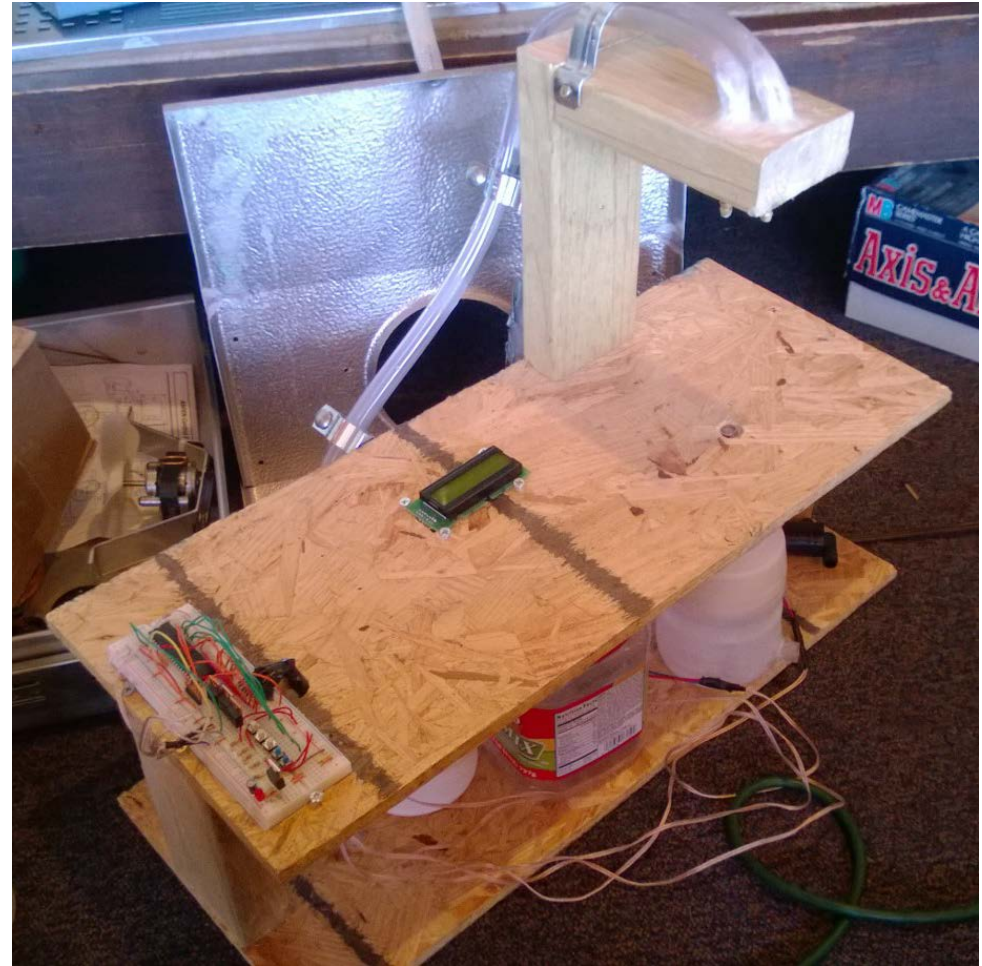
ADVICE

- FOCUS ON THE HARDWARE NOT JUST THE SOFTWARE
- CARDBOARD DOES NOT WORK WELL WITH ICE OR WATER
- START EARLY IN ORDER TO RECEIVE HELP
- BE ROBUST WHEN DEVELOPING A DISPENSING DEVICE, IT IS HARDER THAN YOU MAY THINK TO GET SMALL AMOUNTS AT A TIME

Drink Maker

Will Maethner

Chris Ingram



Objective

- To make drinks with up to three different liquids

Inputs

- Photoelectric sensor sensed when a cup was placed in the ready position

PIC Controller

- Timers
 - Run motor for certain number of ticks to differentiated between drink sizes
- Interrupts
 - To sense when our buttons were pushed

Outputs

- LCD Display
 - Showed your selection to prevent mistakes
- Motors
 - On/off for a certain number of ticks

Results

- It worked as planned as far as coding goes
- Mechanically we had a hard time preventing a few leaks in our project

Advice

- Don't just build the first design, we ended up having to rebuild things because we didn't think of everything the first time. We ended up having to debug the mechanical portion of it opposed to the electronic portion



FORCED CONVECTION BEVERAGE COOLER

Nate Minks & Bruce Coston

OUR GOALS WERE AMBITIOUS.

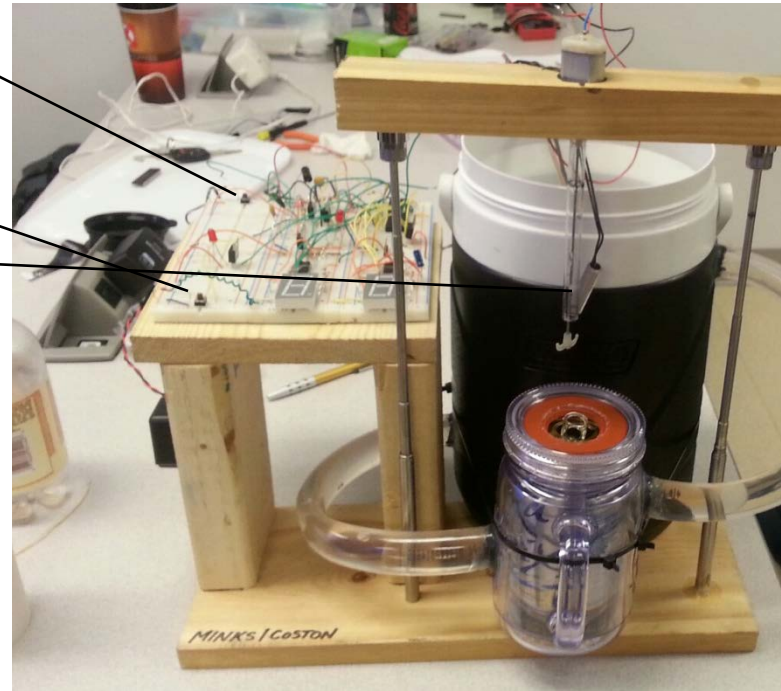
- Cool a canned, room temperature beverage down from room temperature to 40 degrees in under 5 minutes
- Force convection inside and outside of the can
- Coolant used should be ice water
- Use a microcontroller to accomplish this



OVERALL, THERE WERE ONLY 3 INPUTS.

- “Start/Resume” pushbutton
- “Emergency Stop” pushbutton
- RTD temperature sensor

The RTD was a point of difficulty because a non-linear calibration equation had to be made linear by selecting appropriate data and best-fitting a line to the expected temperature range. Also, some math tricks had to be performed to use ints and maintain high resolution.



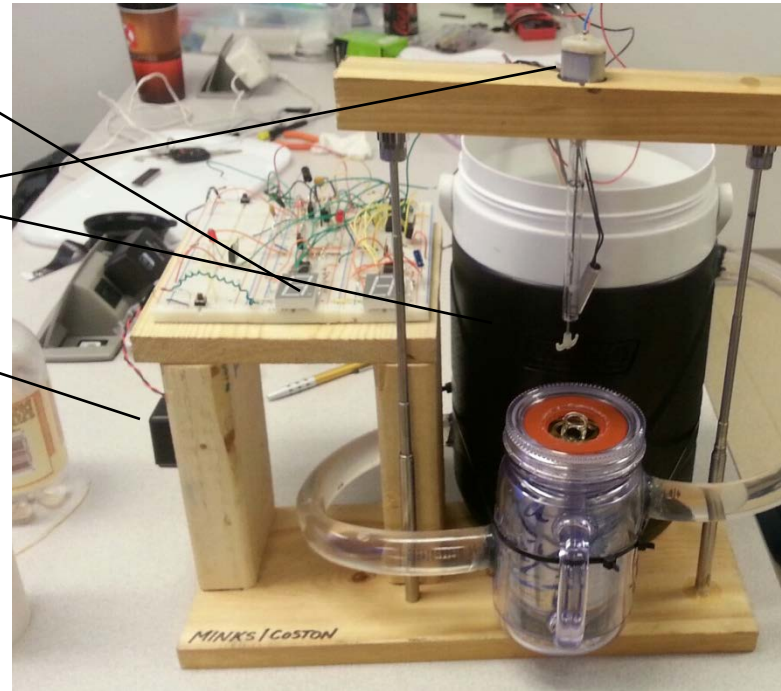
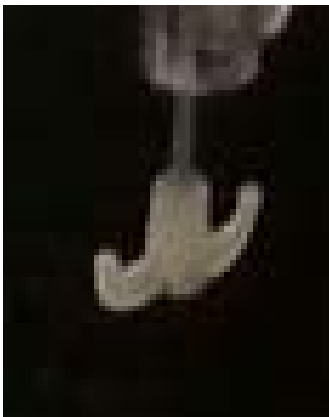
THE PIC CONTROLLED THE DEVICE USING A VARIETY OF SKILLS LEARNED IN CLASS.

- **“Start/Resume” pushbutton**—necessary to enter the loop
- **“Emergency Stop” pushbutton**—controlled on an interrupt, stops the device at any time
- **RTD temperature sensor**—used capacitors to cancel out noise, constantly took an average of multiple readings, when beverage temperature was less than 40-degrees, loop was exited
- **DC motor**—used a unique method of PWM to initially get the motor spinning fast enough to overcome starting friction, then slowed it down almost instantaneously to prevent fizz from shooting everywhere
- **Bilge Pump**—unfortunately not able to be controlled by PIC
- **Siren**—performs a quick series of chirps when temperature has been reached
- **2 seven-segment displays**—displayed current temperature, had to use a bit-shift to have one of these *and* the RTD all be controlled via PORTA.



THE PIC CONTROLLED 5 OUTPUTS.

- 2 seven-segment displays
- Bilge pump
- Siren
- DC motor
 - Attached to spinning blades to force convection inside the can



Overall, 6 airtight seals needed to be maintained to achieve our goals.



SUCCESS!

- ✓ We were able to use the start and stop buttons to control the device. The RTD did correctly sense when the beverage was at our defined 'cold' temperature and therefore shut off the device accordingly.
- ✓ The 7-segment displays correctly displayed the temperature
- ✓ Alarm sounded when 'cold' temperature was reached
- ✓ The motor mixed the drink during the cooling process as planned
- ✓ In fact, **we made it to 39-degrees in under 3 minutes.**

- The pump gave us problems

For some reason that is still a mystery to us, the pump provided some rogue kickback that shut down the PIC whenever we tried to operate it within the circuit (snubbers were implemented). Therefore, we placed an LED in the circuit in place of the pump that was controlled with a MOSFET connected to the PIC. When we did this, we saw the LED operating as the pump should have. We had to separate the pump completely from the circuit and operate it manually with its own power supply. When we did this, the rest of the circuit (RTD, motor, buzzer, 7-segment display, start/stop buttons) operated without a hitch.



SOME ADVICE FOR OUR HEIRS TO THE ME430 THRONE...

- Start working on your project ideas early. Be thinking about them from day one and don't waste time before getting started.
- Test, test, and then test again. Make sure the device is reliable.
- Start by using LEDS in place of larger outputs
- The PICS are very fragile!
- Do research on the components you buy (the pump in our case)
- Start creating your device/project as soon as you have the necessary resources.
 - Even if you don't know how to code your device, you can start building the hardware





THANKS FOR WATCHING!

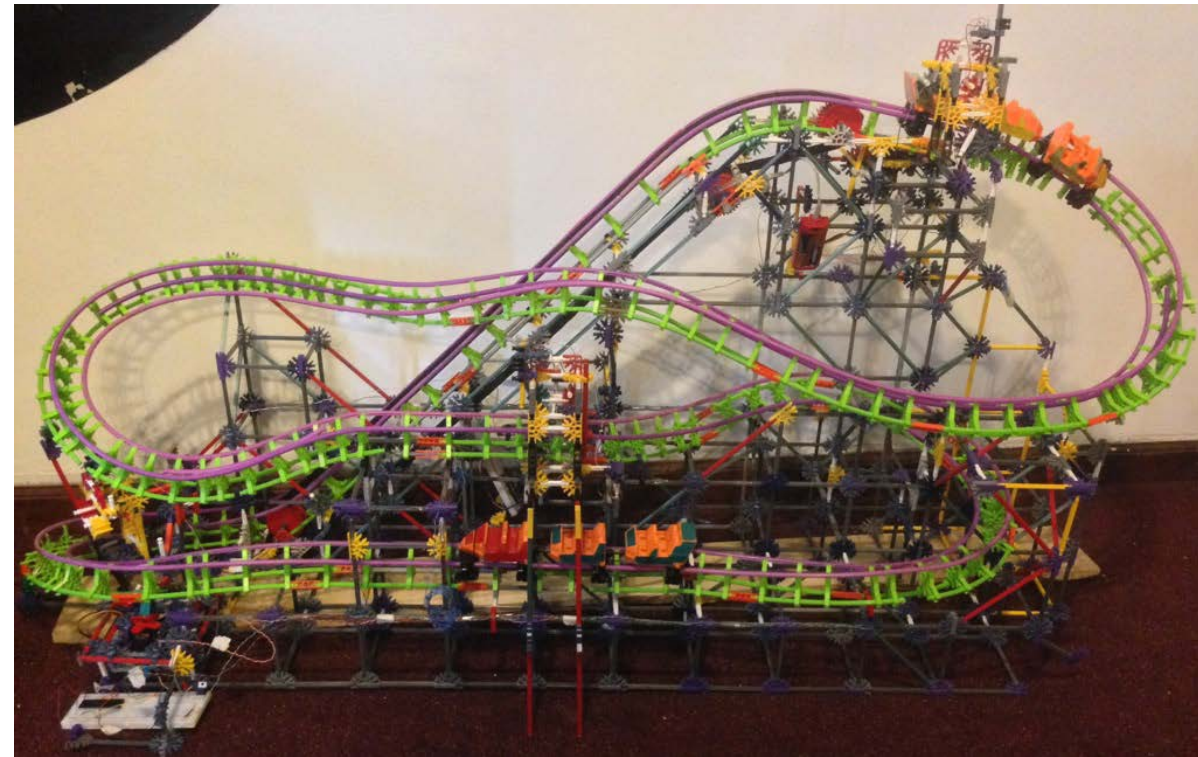
Roller Coaster Block Brakes

Mitchell Feldhake

Luke Reynolds

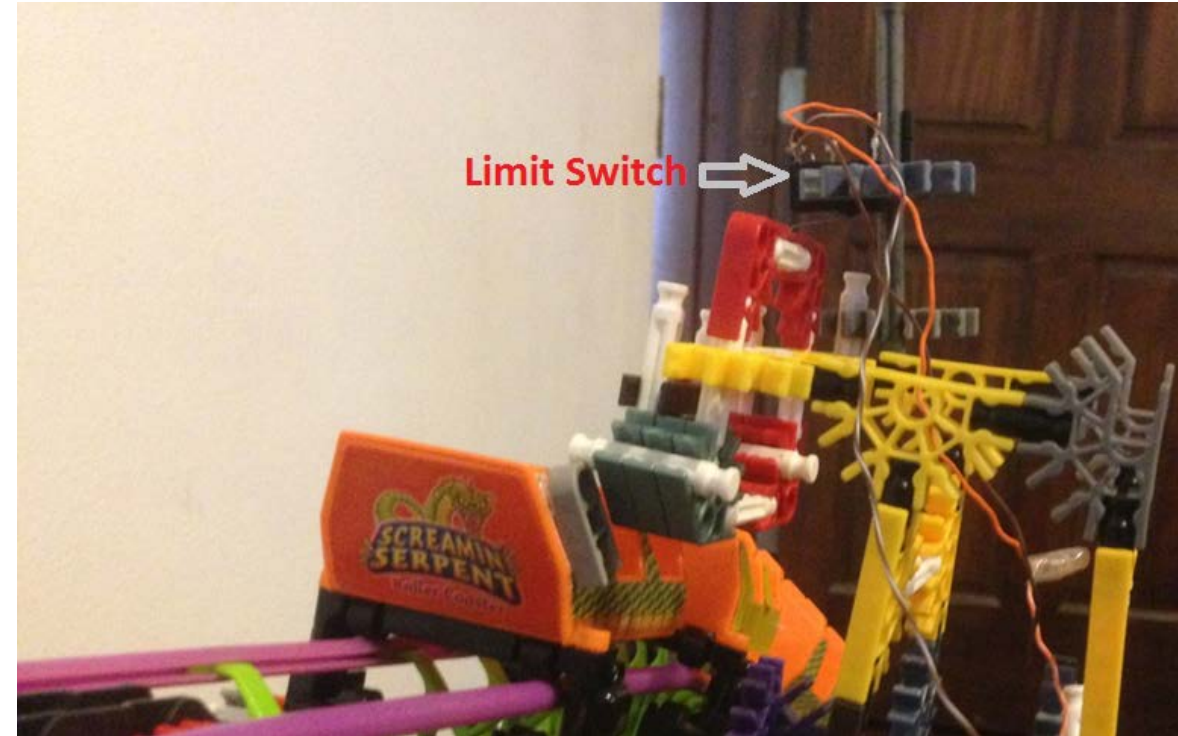
Objective

- To create a functional block brake system on a model rollercoaster:
a system that allows two trains to run on the same track without ever crashing into each other.



Inputs

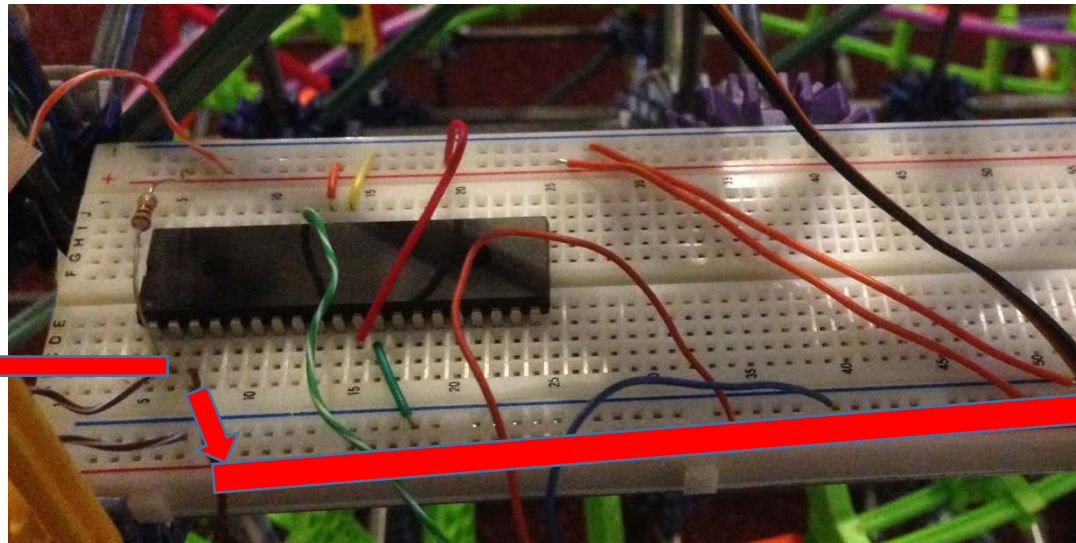
- 3 limit switches (placed alongside track across from brakes)
- As train goes by, it hits a K'nex lever that presses the corresponding switch and activates the outputs.



PIC(s)

- 3 PICs were used – all with the same code.
- When one limit switch was hit, it activated the PIC at it's corresponding block brake, and the PIC at the previous block brake.
 - (The previous PIC opens the brake, the corresponding PIC closes the brake)
- Push Button interrupts were used.

RA0 goes to
corresponding
limit switch



RA1 goes to
previous limit
switch

Outputs

- 3 servo motors (attached to brakes alongside track)
- Holes were drilled into 3 K'nex rods to attach them to the motors in order to move the brakes.
- K'nex pieces took the angular motion of the motor and turned it into linear motion used to move the brake.



Results

- The system worked as planned, with one exception:
 - All the electrical components worked, and the servo motors interfaced beautifully with the brakes, without having any difficulty turning.
 - The limit switches had to be precisely positioned, so that they would be hit hard enough to be pressed, but not hard enough to stop the train from moving.
 - During the demonstration, they only worked about half the time because the switches had gotten unadjusted during transportation.

Advice

- When trying to sense a moving object, a physical sensor may seem more practical and easier than a light sensor, but keep in mind that the sensor could slow the object down.
- When using long wires from different places (like Ethernet cables), make sure to test them with a DMM before attaching them to the project.
- Make sure to transport your projects early so you have time to readjust anything that may have gotten knocked out of place.

Human Powered Vehicle Rear Wheel Steering

Chenchen Shen

Patrick Woolfenden

Objective

The human powered vehicle is building tricycle this year. The tricycle will be able to tilt using a parallelogram linkage mechanism for the rear two wheels. The parallelogram tilting mechanism can be locked vertically so the vehicle can stand upright when coming to a stop or being ridden at slow speeds.

Our system incorporates rear wheel steering that activates while the vehicle is locked vertically. This increases the vehicles turning radius as well as offer new unique driving modes.



Inputs

- A mode selection switch
 - A 2 position switch mounted on our breadboard that allows the rider to select between mirroring the front wheel and joystick mode
- Steering Rate potentiometer
 - To adjust the amount of travel the rear wheels will turn in proportion to the front wheel
- A joystick
 - mounted on the handle bars for independent control of rear wheel steering.
- Steering angle Potentiometer
 - To measure the angle of the front wheel.

Programming

Timer 0 high priority interrupts for all functions (clock cycle: 4 MHz, Prescaler: 4, timer ticks: 5000 → frequency: 50 Hz)

```
OSCCONbits.IRCF2 = 1;
OSCCONbits.IRCF1 = 1;
OSCCONbits.IRCF0 = 0;
OpenTimer0(TIMER_INT_ON & TO_16BIT & TO_SOURCE_INT & TO_PS_1_4);
WriteTimer0(60536);
```

Analog reading for potentiometers

```
SetChanADC(ADC_CH0);
ConvertADC();
while (BusyADC());
potentiometer = ReadADC();
```

Interpolation (mapping) to transfer from analog readings to servo delay time

```
delayTime = (potentiometer - POT_MIN) * delayRange / potRange + DELAY_MIN;
```

Delay time to set angles of the servo motor (RC0)

```
delayTime_hunds = delayTime / 100;
delayTime_tens = (delayTime / 10) % 10 + 1;
PORTCbits.RC0 = 1;
Delay100TCYx(delayTime_hunds);
Delay10TCYx(delayTime_tens);
PORTCbits.RC0 = 0;
```

Outputs

- Signal to servo
 - PWM signal at 50 HZ
- Mode LED
 - An LED mounted on the bread board to show the rider what mode they are in.

Results

- The system worked as planned.
- The servo had enough torque to move the wheels with the weight of a human on the vehicle.

Advice

- Look into a way to consistently delay with a resolution of 1 instruction cycle for higher servo resolution. Possibly using interrupts.
- Try to make the joystick control less responsive. If the joystick is not used smoothly the servo jerks the rear wheels around violently.

Automatic Toilet Paper Dispenser

ME430: Mechatronic Systems



Robert Swartz

Blake Schollaert

Objective

- Design and build an automatic toilet paper dispenser that can be successfully activated with an infrared range motion sensor.
- Allow the user to choose the amount of paper dispensed.
- Give the option for the toilet paper to roll down and roll up back up.

Inputs

- **Inputs to the μ processor:** Analog signal from the Sharp infrared motion sensor.

Infrared Sensor



Outputs

- **Outputs from the μ processor:** The stepping sequence for the stepper motor. If the analog input is between 100 and 400 the motor will spin backwards. If the analog input is above 600 the motor will spin forwards and dispense toilet paper.

Code

- Used simple code
 - No Interrupts
 - No Timers
 - No PWM
- Used **Delays**
 - This slows the rate at which the stepping sequence was sent to the motor.
- Set **threshold** value of the infrared sensor
 - This allows the user to activate the sensor at a reasonable distance.

Results

- Toilet paper dispenser works as planned
- The motor dispenses paper when the pic receives an ADC reading greater than 600, and rolls the paper back up when the ADC reading is between 100 and 400

Advice

- Make sure wires on the seven segment display don't cross each other.
- Make sure the enable line from the H-bridge is not set to low if you want your motor to operate.
- Experiment with infrared motion sensors to determine their sensitivity and activation range.

ME430 Final Project
Andrew Thompson & Ruoyun Li



Put Master® - Automatic Putting Green

Objectives

Put Master[®] is designed to do several things:

- Count the number of strokes it takes the player to hit the ball into the hole
- Count the number of holes the player makes
- Each time the player sets the ball on the tee, the stroke-counting 7-segment LED resets to zero



Inputs and Sensors

Inputs	Sensors	Difficulties
The ball drops in the hole	IR Sensor	The light bouncing around the hole, and allowing the sensor to detect one passing of the ball as one hole.
The ball is put on the tee	Light sensor (CdS Photo Cell)	Sensing the correct amount of light based on the threshold value from the potentiometer.
The club hits the ball	Pressure-Sensing-Resistance pad	Figuring out the wiring of the pad with an op-amp and how best to translate the input into desired format. Originally used analog input, then found that digital worked just fine and was easier.



The Usage of the Parts

	Usage	Difficulties
PIC	Read in analog and digital inputs; translate the inputs into desired outputs	Using different ports for inputs & outputs; finding the proper wires to send the complete signals
Interrupts	To increment the number of strokes when the resistance pad hits the ball.	Changing the Falling Edge to Rising Edge because the voltage input from the sensor increases when it hits the ball.
Timer	To delay and debounce the pressure sensitive pad, and delay in order to count one passing of the ball with the IR Sensor.	Finding the right amount of time to delay in order to count the ball entering the hole one time.
PWM	NA	NA

Outputs

Outputs	How created
Stroke-counting 7-segment LED Display	This was a simple incrementing variable that was incremented every time the interrupt from the pressure sensitive pad was triggered.
Holes-counting 7-segment LED Display	This was another simple incrementing variable that was incremented every time a ball went into the hole. This required some delays and essentially using the debounce technique on the IR Sensor.

Results

The Put Master[®] worked as anticipated. The sensors were able to sense the required inputs and translate them as desired into the outputs. The only problem was that the resistance pad might have been a bit too large and tended to brush the grounded, sometimes adding more to the score. Apart from that and some bad wires, everything worked as intended.

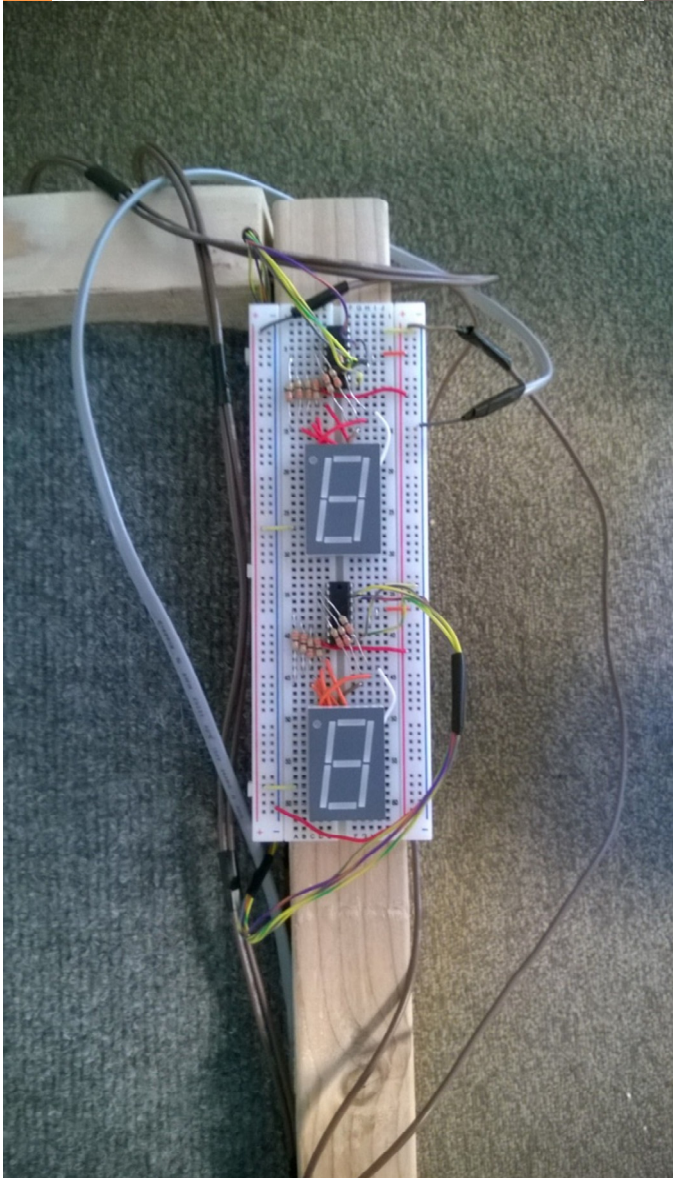


Advice for Future Generations

- Keep project as simple and feasible as you can, it will come down to the last minute, understand that from the start.
- Start building your projects early
- Start thinking about how to program and the functionality you will be using in MPLABx



Pictures



Pictures



Basketball Arcade Game

Ryan Tooley

Adam Westerman



Project Objective

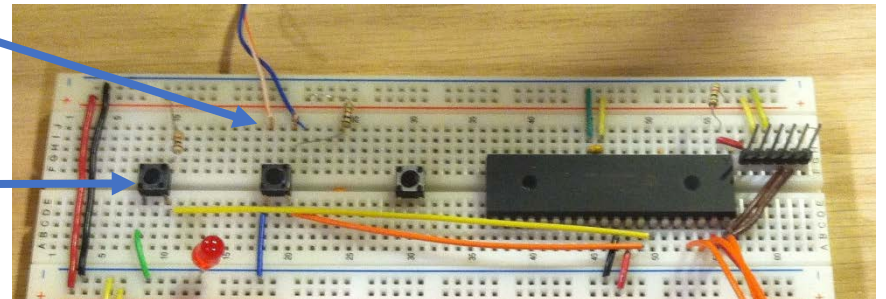
The basketball arcade game allows an individual player to shoot and make as many baskets as possible in a given period of time. The game will keep track of the player's score as well as the time remaining for the given game.

Inputs

The two inputs for our project were the basket made switch and the reset switch. We used a cantilever beam in combination with a mechanical switch to sense when a basket was made by the user. To reset the game we used a simple mechanical switch.



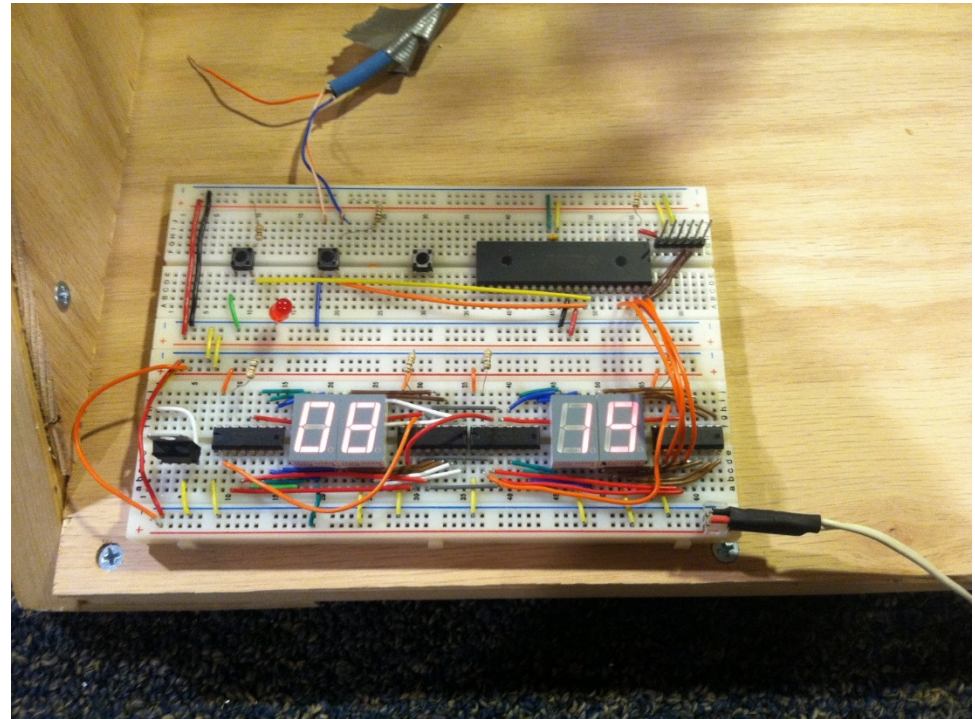
Basket Made



Reset Button

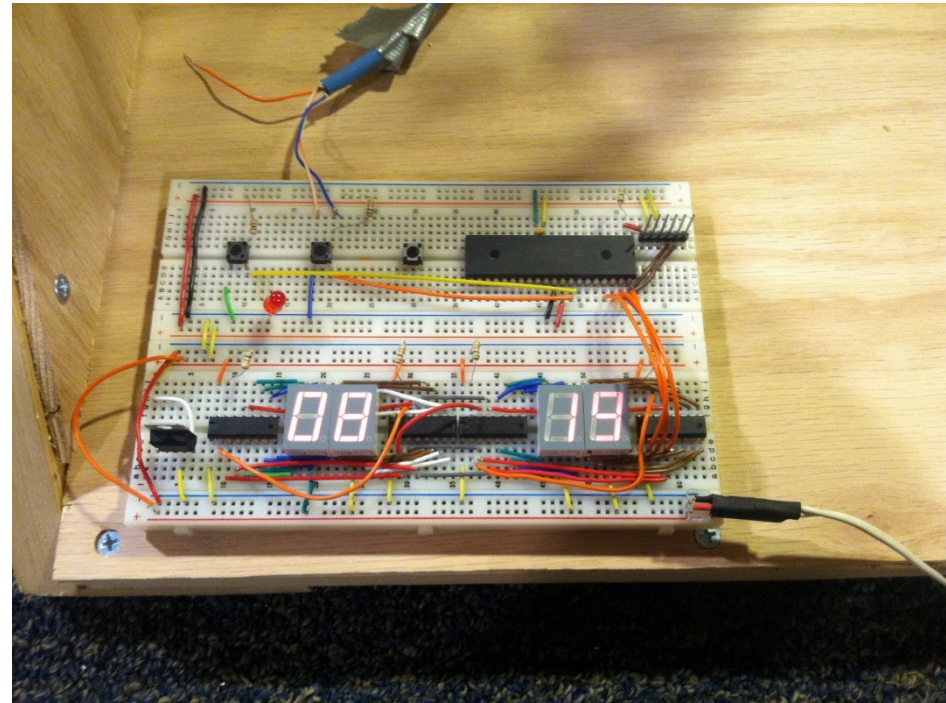
Utilizing the PIC

Timers and interrupts were used within the PIC to create our scoreboard. Timer 0 (with interrupts activated) was used to control the clock, which updated a global variable to “count down” inside the high priority ISR. The RBO interrupt was used in conjunction with our lever-switch to register each basket scored. Again, another global variable was updated inside the service routine to keep track of points scored. Inside the main infinite loop a button was monitored and, when pressed, would reset both the score and the clock internally.



Outputs

Our outputs consisted of four 1" seven-segment displays, creating a two-digit score display and a two-digit clock. The displays were driven by four 8-channel constant current sink drivers (shift registers). The programming for the drivers was slightly more complex, since whenever one display needed to be updated, all four digits needed to be re-shifted into the drivers. It was ultimately a much better solution though, because a) we used 4 output pins on the PIC to control 32 outputs and b) it let us avoid a ton of clutter on the board (the drivers only need one resistor to set the current output for all pins).



Results



The game did function properly, the sensor that detected made could be problematic at times. Sometimes the ball would not hit beam hard enough so a made ball would not register. Other times the ball would bounce and hit the beam registering more than one made (we did debounce the sensor, but sometimes the ball would bounce more than others).



A video demonstrating basketball arcade game is on the right.



Advice to Future Classes

Always have a Plan B! You may have a plan that should work but it would be wise to have a backup course of action in case something goes wrong at some point during the course of the project.

We had planned on using a laser with a photo cell to detect when a basket was made. Fortunately we had ordered a mechanical switch to perform this function when our original parts did not perform as planned.