5-2018

Mechatronic Carnival

Caroline Kutach
Trinity University, ckutach@trinity.edu

Charlotte Liu
Trinity University, zliu1@trinity.edu

Nic Rodriguez
Trinity University, nrodrig4@trinity.edu

Laura Wilson
Trinity University, lwilson2@trinity.edu

Mechatronic Carnival

Group B

Caroline Kutach, Charlotte Liu, Nic Rodriguez, Laura Wilson

ENGR 4367

Dr. Kevin Nickels

May 3, 2018

Table of Contents

Design Summary:

The goal for our design was to create a two-player water gun shooting game. The object of the game is to use a water gun to aim a stream of water at a target and fill up a tank. The player who is able to successfully fill up their tank first is the winner.

The design consists of two water guns that are powered by a pump. They are used to shoot at a target in the backboard of the device. As the tank fills up, LEDs light up to indicate each player's progress. When a tank is full, all of the LEDs will be on, and music will play. Also, when one player's tank is full the game is over meaning that both player tanks drain, and the pump turns off.

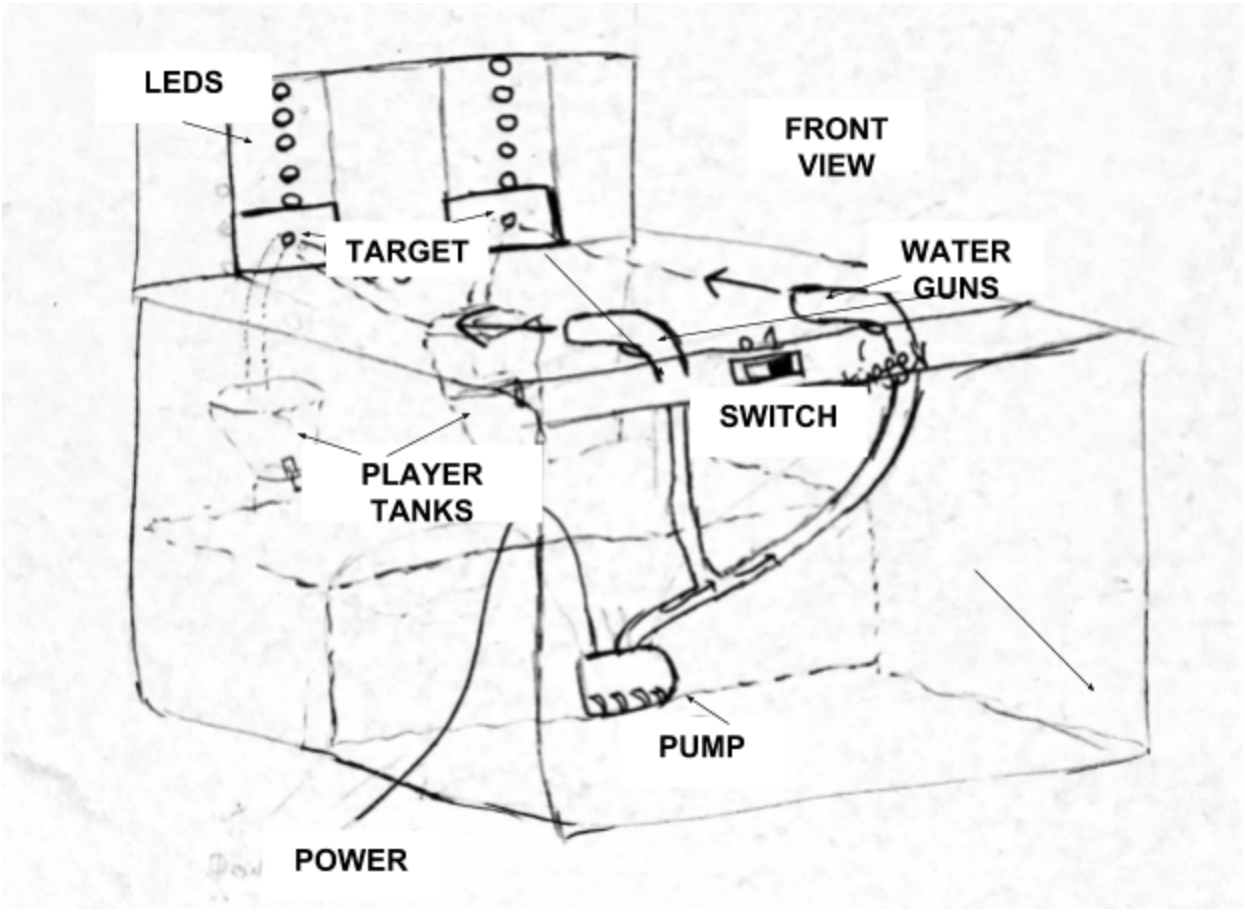


Figure 1. Preliminary sketch of front view.

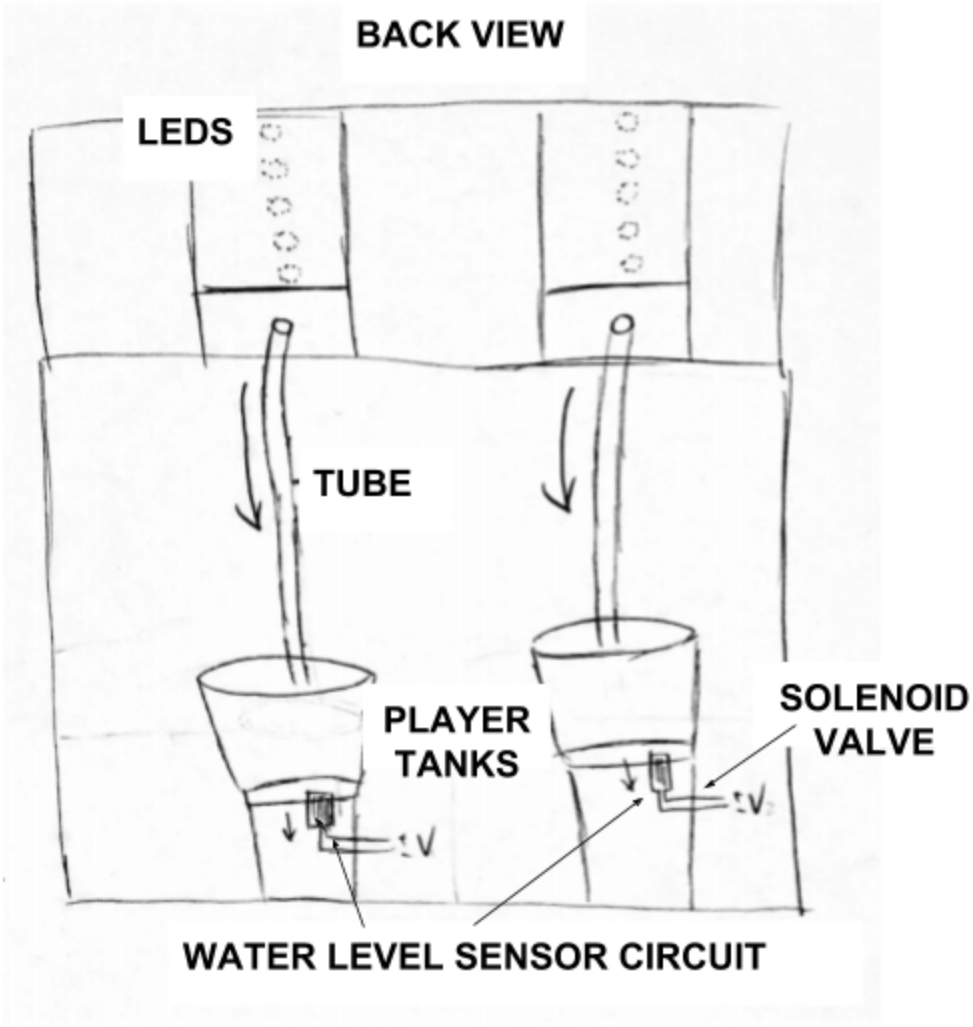


Figure 2. Preliminary sketch of rear view.

System Details:

Before starting the game, the pump must be connected to 110VAC wall power. Excluding the pump, the design is powered by a 12V DC battery which is also stepped down to lower voltage levels. A button placed at the top of the device starts the game. When the button has been pressed the music plays, the pump turns on, and water shoots from the guns, signaling the start of the game.

In order for the water to shoot a maximum distance, we designed and 3D-printed our own water guns, shown in Figure 3. We applied basic fluid dynamics concepts in designing the guns in order to obtain a high velocity flow so that the water streams could reach the targets.

As the users begin to fill up the tanks, LEDs light up in accordance with the water level sensors. When the water level reaches each new level of the tank, a sensor causes an LED to light up, showing the user their progress. When a tank is entirely full, all of the LEDs are all turned on, indicating that a player has won. The highest water level sensor was wired to communicate with the PIC so that when the highest water level is obtained, the pump turns off, the solenoids open to drain the tanks, and the LEDs turn off. The tanks are located on the back of our device, shown in Figure 4. The water level sensing mechanism is a digital circuit and does not rely on instructions from the PIC.

When the game is over, solenoid valves attached to the bottom of each tank open to release the water. Because the valves required 12V to operate, solid state relays were used to interface the 12V source and the PIC, isolating the PIC wiring from the high voltage. Figure 5 shows how the valves were attached to the tanks.

While the tanks are in the draining mode the water pump controlling the flow of water should be turned off to prevent excess water from spraying out. To accomplish this task, one of the wires connecting the pump to power was spliced and a large normally-open mechanical relay was placed in series with the spliced wire. When the relay is closed the pump is powered, providing flowing water for the system, and when it is open the pump would turn off. This relay was used to control the water flow throughout the game and was digitally controlled via a PIC-MOSFET interface.

The logic was programmed using PICBasic, with a PIC16F88 as the microcontroller. The components were programmed to interact with each other sequentially, as shown in the software flowcharts in Figures 6 and 7.

Figure 3: Front view: 3D-printed water guns are connected to a pump that is hidden under the wood casing in an ice chest full of water. There is a target and LED progress bar for each player. The drain at the far end of the tank is used to collect the water that does not go into the targets.

Figure 4. Back view: water bottles collect the water that the users shoot into the targets. The water level sensor wires are fixed at five specific heights in each bottle.

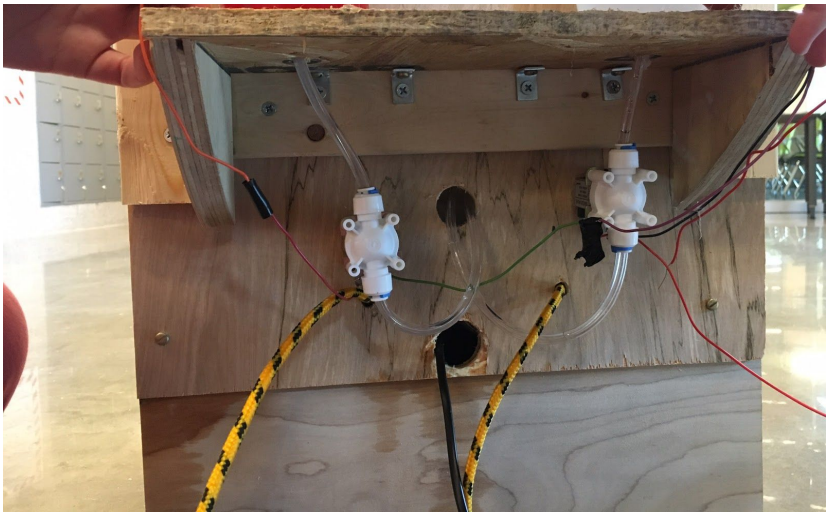


Figure 5. Solenoid valves drain the tanks automatically at the end of each game.. A single exit for the pump wall plug is shown at the bottom. The yellow rope is one of two handles that make transporting the cabinet safe and easy.

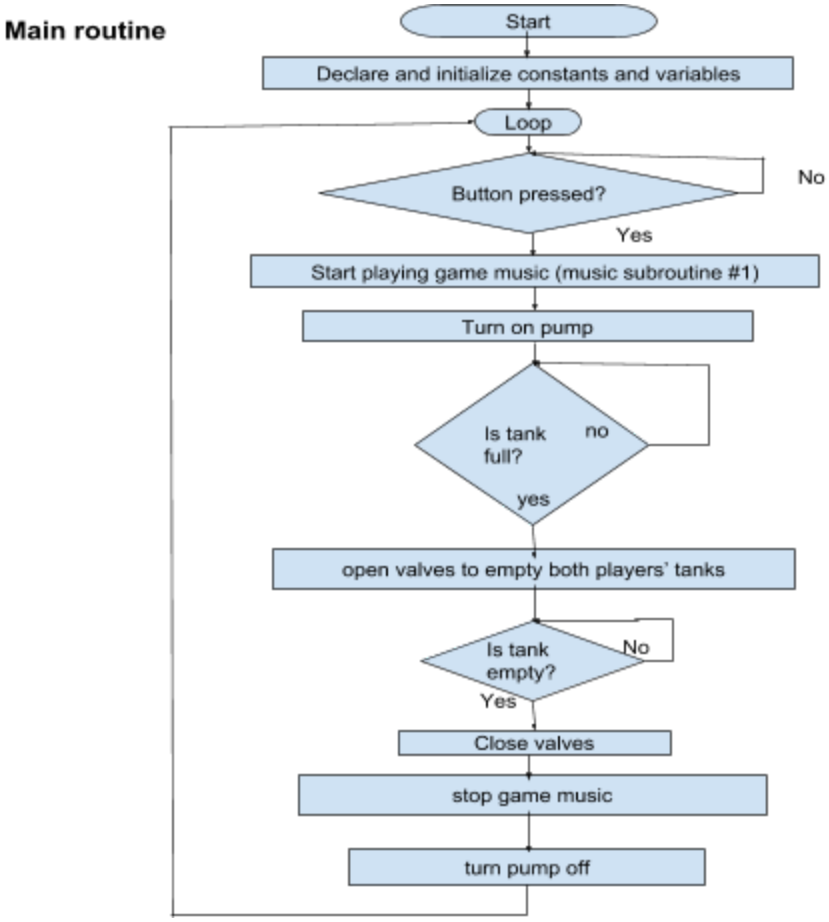
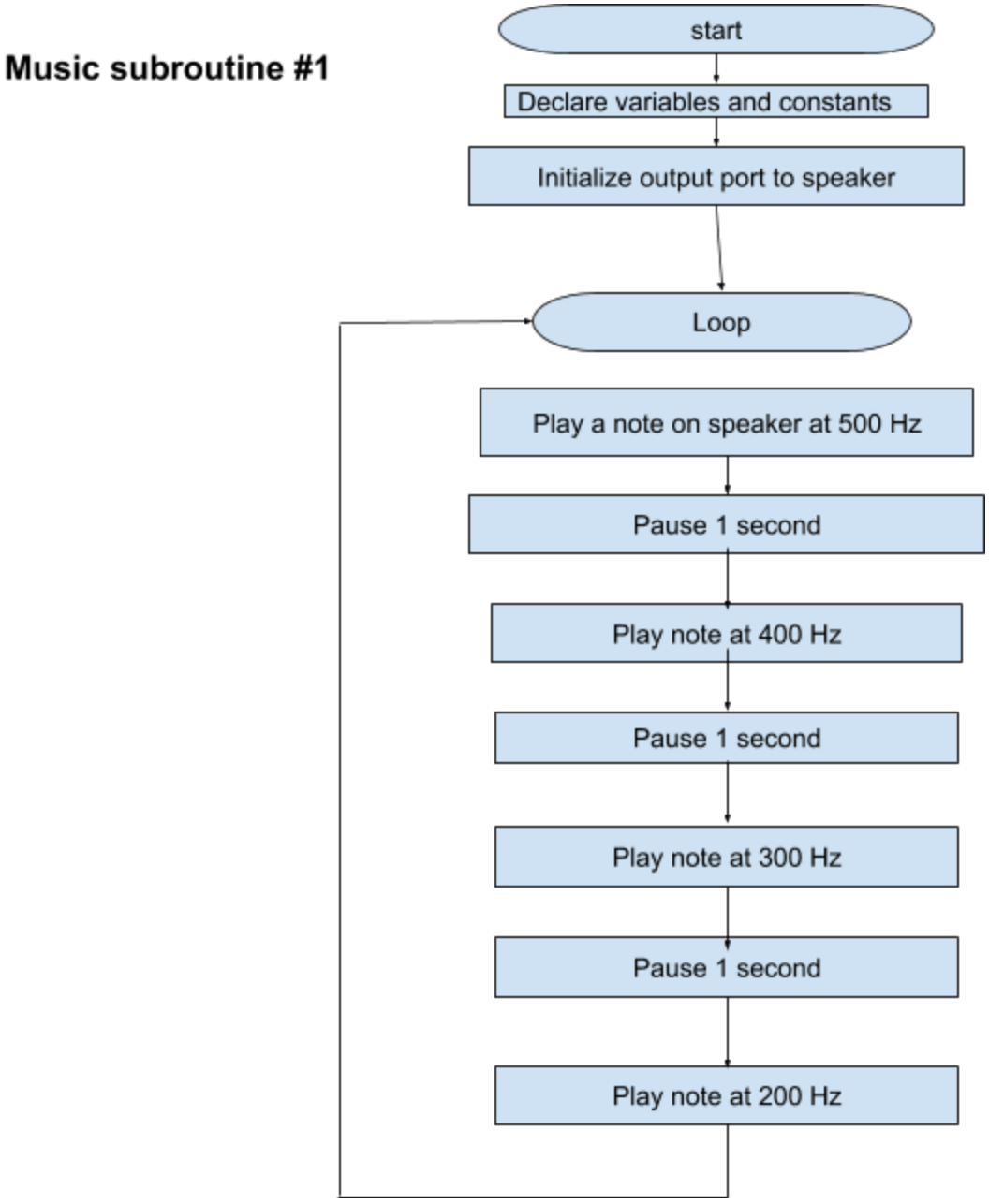


Figure 6. Software flow chart



Design Evaluation:

The output display was two sets of four LEDs to indicate each player's progress. This functional element was successful in that the digital circuit accurately energized the proper LEDs. However, the final circuit organization and soldering impacted the reliability of the LEDs and water sensors. We give the output display category a rating of 16/20 because the LED setup was the most meaningful and simple output display we could have chosen. The digital water level circuit directly communicated with the LEDs, bypassing the PIC.

The audio output category (14/20) was a piezo speaker and code that made use of the PIC BASIC command FREQOUT. We did not use either during homework assignments or lab exercises in this class, but our implementation was fairly straightforward. Again, due to the fact that we did not allot enough time for careful and modular building and testing, this part of the design was not reliably working on the final prototype. We adjust the rating down to 12/20 to account for this.

The manual user input (5/20) we implemented was a game reset button. We had intended for the audio to be two separate short pieces of music to indicate the beginning and end of the game. However, we were unsuccessful on the software side. The plan was to implement a separate subroutine for each of the two cases, but the FREQOUT command did not behave as expected when used with GOSUB. In the end, the code sent a single melody to the piezo, so this category was not very involved in the final prototype. Also, it did not always work reliably for the previously mentioned wiring and organization reasons.

The automatic sensor component (19/20) was a digital circuit consisting of BJTS and resistors. The automatic sensing occurred without PIC control. This circuit energized the LEDs at the appropriate times and sent a signal to the PIC once the highest water level was reached. We arrived at this digital circuit after further simplifying an example that used NAND gates. The combination of BJTs and resistors was a robust and extremely low cost solution. As long as the circuit was powered with 5V, the water level sensor functioned. The only caveat was that the sensors are sensitive to splashing. The next iteration of this design would make the sensor less sensitive to splashing during the game.

For the actuators category (17/20), we used two solenoids and a 110 VAC pump (meant for fish tanks and fountains). Additionally, we designed and 3D-printed custom water guns to attach to the pump tubing. The guns work very well because the inside diameter of the gun exit is smaller than the entrance diameter, making the fluid velocity increase. The guns project the water at least three feet horizontally. The solenoids drain the water correctly, albeit more slowly than we would have preferred.

For processing (8/20) we used s programmed logic in PIC BASIC. The code reliably worked on the breadboard, but did not use any features that we had not already learned in the course. The logic did not reliably work on the combined prototype, although that was most likely due to incorrect soldering, rather than the code.

Qualitative Adjustments:

The final prototype of the game cabinet was well-constructed out of plywood, a sturdy plastic ice chest, and featured rope handles on both ends so that the device could be carried safely by two people. Hot glue was only used to make gaps and connections more watertight. The majority of the wood components (walls, shelves, etc.) were securely fastened with screws. The game cabinet also features thin side walls to prevent the water from spraying outside of the device. We anticipate a +7 adjustment in this category.

We anticipate +2 for a fairly frugal budget. Without including the prices of readily available plywood and screws, the budget is less than $30.

We anticipate +5 for apparent level of effort. The water guns are the fourth iteration we chose after trial and error. We printed the final models out of a higher quality PLA. The cabinet was nicely constructed and with some changes to the rear side for safety and waterproofing could be marketable to consumers.

We anticipate -7 for performance during the demonstration. We were able to get the water level circuit, output display, and actuators to work if manually energized (in recorded video). However, the PIC control was never functioning during the demonstration or video.

We also anticipate -6 for the assembly/safety due to the unorganized and somewhat unprotected wiring and power sources in the prototype. We placed the circuit behind the backboard of the game cabinet and set it up so that the wires could be tucked into a plastic pencil secured to the board (away from water). In the demonstration though, many wires were tangled and coming out of the pencil box.

We expect no deduction for expensive controllers or inconvenient power sources because we only used two PIC16F88s and the user only has to plug in one power cord into a wall outlet (the 12V battery inside the cabinet).

If another group was to recreate or modify this design, the main changes would probably be water-proofing the wooden and electronic components and upgrading the Piezo to a standalone desktop speaker.

Partial Parts List:

Part name	Description	Model number	Source	Price
normally-closed solenoids	12 V, ¼" inlet, no water pressure required, 4.8 W, 3.2 oz	ROCON-341	Digiten	$7.49 each
submersible water pump	24 W, 10V, 800 GPH, ½" outlet, 9.8 ft, 2.01 lbs, adjustable flow rate	B071RVPNQL	Pedy	$19.99
mechanical relay	5V activation 20A N.O. 10A N.C. 240VAV	T90N5D12-5	Electronics Shop	N/A

Lessons Learned:

The first problem that we faced was finding a sensor capable of measuring specific water levels. We thought that we would need to use a more expensive pre-packaged sensor, such as a photoemitter, in order to detect specific water levels. However, we expanded our knowledge past the sensors that we learned in class to construct an inexpensive device out of bipolar junction transistors and resistors.

Another component issue that we encountered was determining the specifications for our pump. We needed the pump to be able to shoot water for two feet when piping and the water guns were attached. However, when buying the first fish-tank pump, we did not consider the pump flow rate specifications. We were mostly concerned that we could power the pump using wall power. We assumed that since we were using the amount of water used in a fish-tank that this pump would perform to our expectations. However, the initial pump did not shoot water far enough to reach a target, so we had to order a second one which had a flow rate large enough to meet our requirements.

Another problem that we faced was safely using AC power. The 110 VAC pump relay presented dangerous conditions when it was not insulated. It is important to ensure that the relay is not short circuited in order to preserve components and avoid user harm. This relay presented even more of a risk when it was in a wet environment. We had to protect the relay from getting wet by covering it and carefully placing it in case the water gun was aimed directly at it.

The most important lesson that we learned was the importance of integrating all of our components onto a protoboard in modules. On separate breadboards our electrical components all functioned according to design, but we implemented them all together onto our final board rather than piece by piece. This made troubleshooting incredibly difficult, and made the layout of our circuitry very unorganized.

The unorganized layout was also due to late planning. We spent the majority of our time on the electronic functionality, and when it was time to implement them together, we had not considered how they would interact with water. Although we placed the electronics on the back side of the game so that water would not shoot directly at them, leaking in parts of the wood required us to quickly find a method to insulate the circuit from water.

Throughout this project, we should have more thoroughly documented our design process. Periodically recording our observations and taking videos of our functional components could have helped us later during troubleshooting as well as during demonstrations to prove that the device was functional.

References:

BrightHub Engineering
https://www.brighthubengineering.com/diy-electronics-devices/75673-build-your-own-home-made-water-level-meter/

Appendix:

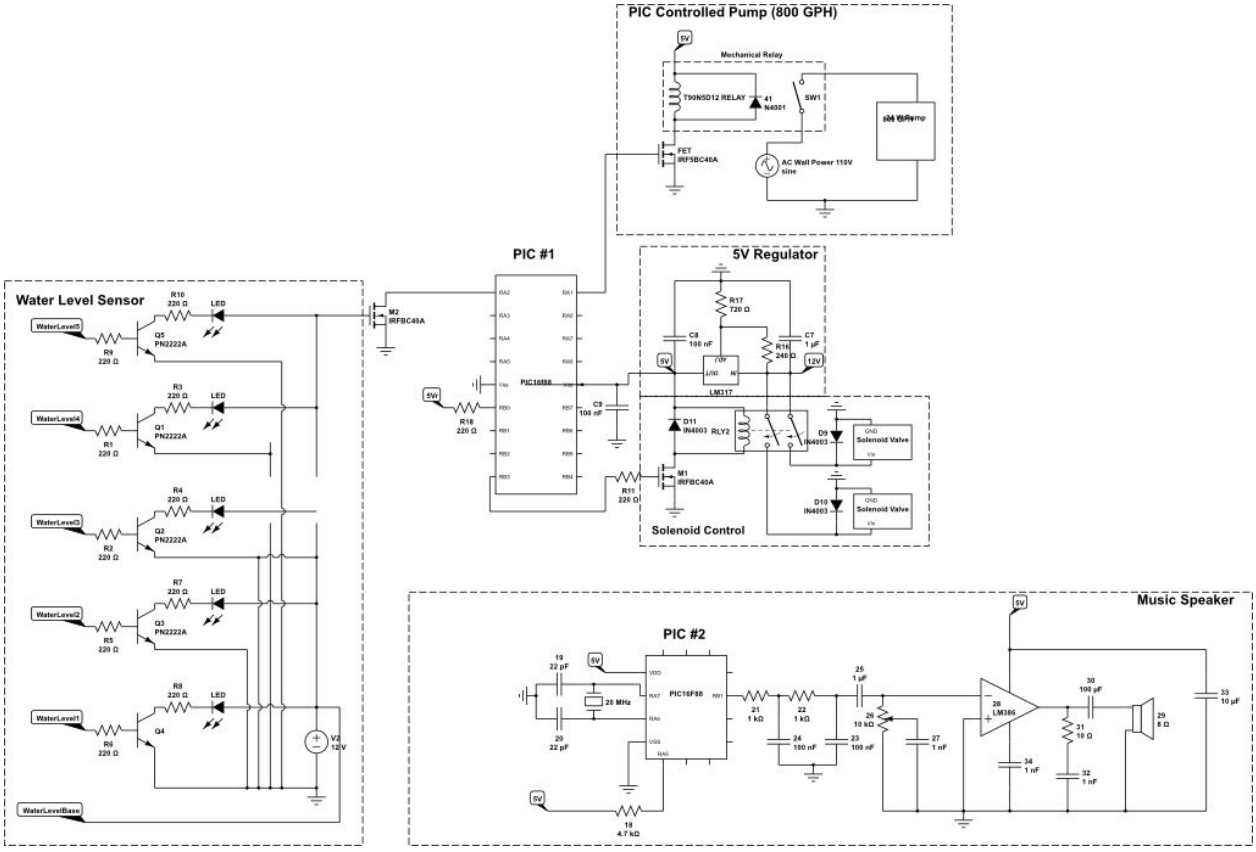


Figure A1. The detailed wiring diagrams of the design project including water level sensor diagram, music speaker diagram, PIC controlled pump diagram, and the 5V regulator diagram.

Figure A2. Designed 3D-printed creo prototype for the water gun.

```
' If player presses button, the game begins (pump turns on, music plays). When
' either player successfully fills up their tank completely, the game is over
'(pumps turn off, the solenoids open).

'Define configs
#CONFIG
    __CONFIG _CONFIG1, _INTRC_IO & _PWRTE_ON & _MCLR_OFF & _LVP_OFF
#endconfig

'Osccon setup
define OSC 8
OSCCON.4 = 1
OSCCON.5 = 1
OSCCON.6 = 1
ansel = 0

'Define variables and constants
testLED var PORTB.2
output testLED

buttonMemory var bit
winnerFound var bit

'Define variable names for input pins
sensorInput var PORTA.2
input sensorInput

buttonTrigger var PORTA.1
input buttonTrigger

'Define variable names for output pins
solenoidValves var PORTB.3
output solenoidValves

pump var PORTA.0
output pump

talkToSpeaker var PORTB.4
output talkToSpeaker
```

```
'Make sure the solenoid, speaker and pump are low,
'and the sensor, button and LED are high,
'initialize the input and output states
low solenoidValves
high sensorInput
low talkToSpeaker
high buttonTrigger
low pump
high testLED
buttonMemory = 0
winnerFound = 0

'Main loop structure
main:
    while(1)
        gosub check_button          'wait for the button to be pressed
        gosub check_water_level     'wait for a certain water level reached
        gosub game_reset            'wait for the game to be reset
    wend

'Checks if the water level reached the sensor
check_water_level:
    if sensorInput == 0 then    'Opens Valve when the player tank reaches its
        winnerFound = 1         'highest seneor level, game over
    endif
    if sensorInput == 1 then    'Game continues until the sensorInput is 0
        winnerFound =0
    endif

check_button:
    if buttonTrigger ==0 then    'Mannual input, start game
        buttonMemory =1
        high pump
        high talkToSpeaker
    endif
    if buttonTrigger ==1 then    'when the button is released, reset button state
        buttonMemory =0
    endif
game_reset:
    if winnerFound == 1 then     'game over, empty tanks, turn off pump,
        high solenoidValves      'and wait for player tanks to drain completely
        low pump
        pause 60000
    endif
```

Figure A3. MicroCode for the overall system including comments.