# Introduction

This design is ½ of a demonstration for Electronica 2018.

This document describes the main controller. The remote-control design is described in a separate document.

The main controller consists of 2 water tanks that are controlled and measured by a PSoC 6 WiFi/Bluetooth kit. It also has LEDs, a speaker, and a UART interface. There will two teams of three people each (chosen randomly from the audience). The goal is for each team to try to fill their water tank first. To do this, they use remote controls that we will provide containing another PSoC 6 WiFi/Bluetooth module, CapSense, OLED Display, and LEDs. The contestants will use a CapSense slider on their remote to swipe left to try to fill the left tank or swipe right to try to fill the right tank. Messages will be sent to the main controller via either WiFi (using AWS) or BLE.

# Hardware

### List

1. CY8CKIT-062 PSoC 6 Pioneer WiFi Kit
2. Custom Shield with:
   1. Dual Motor Driver (TB6612FNG)
   2. Dual LED Control for WS2812 RGB LEDs
   3. Amplifier and Speaker Output
   4. Dual connectors for Liquid Level Sensing
   5. Start Button
3. CY8CKIT-022 (2) to do liquid level sensing on two bottles.
   1. Note: The sensors and bottles from this kit are used but not the shield.
4. Submersible Pumps (2)
   1. <https://www.amazon.com/Mavel-Star-Submersible-Fountain-Upgraded/dp/B0713T9PRP/ref=pd_lpo_vtph_200_lp_t_2?_encoding=UTF8&refRID=CAMHGDA9695DQ80AY3AR&dpID=41Da%252BTqs1VL&preST=_SY300_QL70_&dpSrc=detail&th=1>
   2. These need to have their connectors cut off and wires stripped and tinned so that they can be connected to the shield.
5. Plastic Tubing
   1. <https://www.amazon.com/dp/B000E62TCC/ref=twister_B07GZVYNXF?_encoding=UTF8&th=1>
6. Bucket (4L or larger)
7. Duct Tape (or is it Duck Tape?)
8. (Optional) Base to connect various parts of the setup.
9. (Optional) Caps for the bottles (2) with holes for the tubing drilled in them.

### Assembly

To assemble the demonstration:

1. Attach 2 liquid level sensors to 2 bottles from the CY8CKIT-022.
2. Connect the custom shield to the CY8CKIT-062.
3. Connect the 2 liquid level sensors/bottles to the custom shield.
4. Place the entire assembly on the base and attach if necessary/desired.
5. Place the bucket near the bottles (on the floor under the table is probably OK) and fill with at least 3L of water.
6. Connect plastic tubing to each of the 2 pumps. The tubing goes on the outlet port which is on the side (opposite the power cord).
7. Connect the pump power leads to the custom shield.
8. Place the pumps inside the bucket.
9. Connect the tubes from the pumps to the bottles – attaching them right at the top of the bottles is best – if they are inside the bottles you can get siphoning back into the bucket.
   1. We might want to use caps on the bottles with a hole drilled in them to just fit the tubing. GJL can collect caps that fit the bottles from the water level sensing kit.

# Firmware

### Startup

Upon powerup or reset, the firmware will be in a state that is ready to start the game. It will:

1. Connect to WiFi.
2. Connect to AWS and Subscribe to MQTT game messages.
   1. The broker name is: "amk6m51qrxr2u.iot.us-east-1.amazonaws.com"
   2. Messages will be received from the MQTT topic "PumpAWS".
   3. We will use the AWS account wiciedwifi101.
3. Act as a BLE Central and scan for BLE devices with specific custom manufacturer data (vendor ID and product ID).
4. Allow up to 6 BLE devices to connect as Peripherals.
5. Act as a GATT Server. The server will have:
   1. One Custom Service Containing the following Characteristics:
      1. WaterLevelLeftBLE
      2. WaterLevelRightBLE
      3. PumpLeftBLE
      4. PumpRightBLE
   2. The first 2 Characteristics will be Readable by the Clients and the last two will be Writable.
6. Display pertinent information on a UART terminal. For example, messages regarding WiFi connection, MQTT connection, BLE, etc. to make sure things are running properly.
7. After initialization, measure the water level in each tank (presumably will be 0) and:
   1. Publish the values to the MQTT broker. See the Game Operation details below for the shadow topic to use.
   2. Update the values in the WaterLevelLeftBLE and WaterLevelRight Characteristics.
8. Wait for user input to start the game (using a mechanical button or UART command).

### Game Operation

A mechanical button or UART command will be used to start game operation. Once the game starts the firmware will:

1. Play a sound for a short time (fight bell).
2. Monitor water level in both tanks. LEDs will be lit to represent the amount of water in each tank.
3. Look at MQTT messages received and increment the appropriate counter for each message it receives. The counter will increment by the amount of the value passed by the message. Messages will be JSON and will indicate which counter to increment and by how much.
   * 1. Example message for the left pump counter is: {"Left" : 2}
     2. Example message for the right pump counter is: {"Right" : 3}
4. Increment the appropriate counters when the PumpLeftBLE and PumpRightBLE Characteristics are written. The counter will increment by the amount of the value written to the Characteristic.
   1. Note that the value doesn't need to be stored anywhere – the GATT write callback just needs to increment the appropriate counter.
5. Activate the appropriate pump when that pump's counter is greater than 1.
   1. The firmware will use a varying PWM duty cycle in which it pumps faster for larger values in the counter. This will be determined through experimentation for good game play.
   2. The firmware will decrement the counter on a periodic basis until it reaches 0 at which point the pump will be stopped. The decrementing rate will also be determined through experimentation for good game play.
6. Publish MQTT messages to the "Electonica2018" Thing Shadow with water levels. The topic name will be *$aws/things/Electronica2018/shadow/update*. This can be done periodically (e.g. every 250ms) or just when the water level crosses a threshold (every 5%). An example message is:
   1. {"state" : {"reported" : {" WaterLevelLeftAWS" : 20.0, " WaterLevelRightAWS " : 25.0}}}
   2. GJL will create the *Thing* named Electronica2018 and will provide the necessary certificates.
7. Update the water levels in the WaterLevelLeftBLE and WaterLevelRightBLE Characteristics in the GATT database. This should be done on the same frequency as the MQTT publish messages (i.e. either on a periodic time bases or when water level crosses a threshold).
8. Display water level messages on the UART – use the same frequency as MQTT/BLE.
9. When one tank is full (>95%), end the game by doing the following:
   1. Shut off both pumps.
   2. Play an "end of game sound".
   3. Flash the LEDs rapidly on/off for the side that won the game.
   4. Display an appropriate message on the UART.

To restart the game, the kit will be reset.

# Questions:

1. Do we want to use Micrium to show the two tank levels on the screen? This is done in the water level sensing kit example project for a single bottle. Can we leverage that?
2. Which of the 2 liquid level sensors do we want to use? There is one with 2 sensors (backgammon style) and one with 12 sensors. Presumably the 12 sensor one is more accurate, so we should probably use that one.