# Introduction

This design is ½ of a demonstration for Electronica 2018.

This document describes the remote control. The main controller 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. Custom Board with:
   1. PSoC 6 WiFi/BLE module
   2. OLED Screen
   3. 2 CapSense buttons with 2 LEDs
   4. 1 CapSense slider with 1 LED
   5. Power switch and LED
   6. Battery charger and LED
   7. PSoC 5LP based KitProg and USB connector
   8. Motion Sensor
2. A rechargeable LiPo battery
3. A transparent acrylic case
4. A sticker with the Mouser logo and a QR code that goes to a landing page with information about the remote control (schematic, BOM, block diagram), ARH blog about the project, videos from Electronica, etc.

# Firmware

### Startup

Upon powerup or reset, the firmware will do system initialization, print a welcome message on the OLED screen and then wait for one of the 2 CapSense buttons (labeled Bluetooth and WiFi) to be pressed.

### WiFi

If the user presses the WiFi button, the kit will:

1. Blink the LED next to the WiFi button and show appropriate connection status messages on the OLED.
2. Connect to a WiFi network (network name/password will be hard-coded in the Firmware)
3. Setup an MQTT connection to AWS
   1. The broker name is "amk6m51qrxr2u.iot.us-east-1.amazonaws.com".
4. Once the connection is setup, the LED next to the WiFi button will stay on.
5. The kit will subscribe to the MQTT shadow topic that indicates water levels.
   1. The topic name is *$aws/things/Electronica2018/shadow/update/documents*
6. When a message from the broker is received, the two water levels will be extracted from the message and will then be displayed on the OLED.
   1. An example message is:
      1. {"state" : {"reported" : {"WaterLevelLeftAWS" : 20.0, "WaterLevelRightAWS" : 25.0}}}
7. At this point, the kit will wait for CapSense slider swipes.
8. For each swipe, the kit will publish an MQTT message to the topic "PumpAWS". The message will be JSON and will indicate the pump to operate and by how much. For example:
   1. Example Left Pump message is: {"Left" : 2}
   2. Example Right Pump message is: {"Right" : 3}
   3. The value sent will be proportional to how large a swipe was performed by the user. The scale will be determined by experimentation to get good game play action.
9. To restart the game, the kit will be power cycled.

### BLE

If the user presses the Bluetooth button, the kit will:

1. Blink the LED next to the Bluetooth button and show appropriate connection status messages on the OLED.
2. Advertise a GAP Peripheral with a specific custom manufacturer data (vendor ID and product ID).
3. Once the GAP Central (the Main Controller) connects, discover Services.
   1. The Remote Control will be a GATT Client
   2. The Central is a GATT Server with a custom Service containing the following Characteristics:
      1. WaterLevelLeftBLE
      2. WaterLevelRightBLE
      3. PumpLeftBLE
      4. PumpRightBLE
   3. The first 2 Characteristics are readable by the Client and the last two are writable.
4. Once the connection is setup, the LED next to the Bluetooth button will stay on.
5. The Client will periodically read the WaterLevelLeftBLE and WaterLevelRightBLE Characteristics from the Server and will display their values on the OLED screen.
6. At this point, the kit will wait for CapSense slider swipes.
7. For each swipe, the kit will send a value to the PumpLeftBLE or PumpRightBLE Characteristic on the server.
   1. The Characteristic written will depend on the swipe direction.
   2. The value sent will be proportional to how large a swipe was performed by the user. The scale will be determined by experimentation to get good game play action.
8. To restart the game, the kit will be power cycled.

# Questions:

1. Instead of polling for the water levels, we could just subscribe for notifications of changes. In this case, the main controller would have to accept notification requests and send notifications to 6 clients. That may be more difficult to manage.