Ball Balancing Table

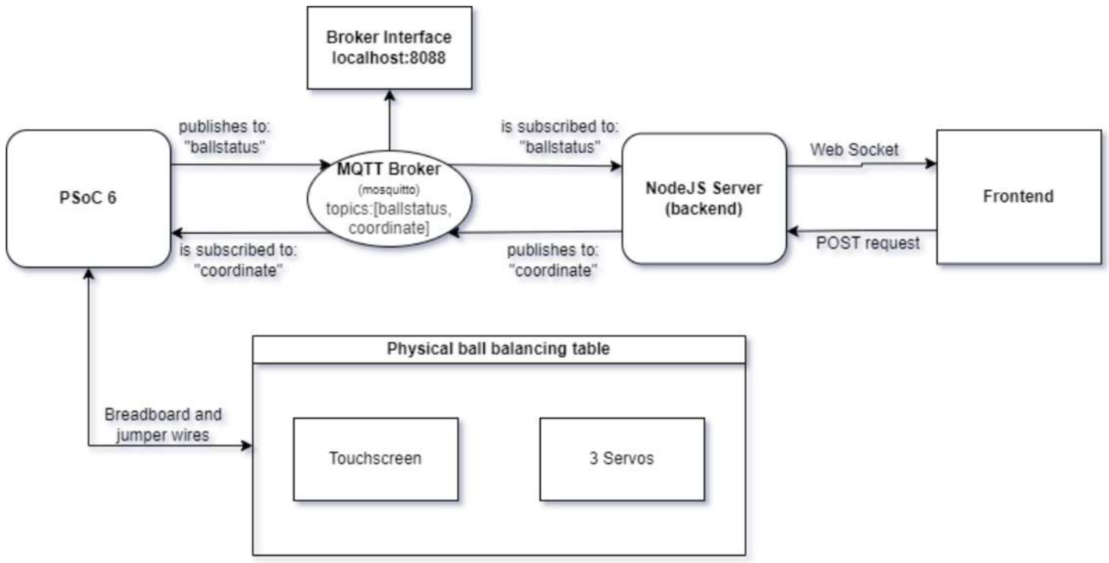
Goal

The goal of this project is to have the system balance a ball on a table. The user gets an interface on which the position off the ball is shown by a marker. The user would then be able to interact with this interface by clicking on the rectangle representing the balancing area of the table. The table will then balance the ball to the position the user has clicked on.

Setup

A PSoC 6 is used as the main controller of the system. It is connected over Wi-Fi to a MQTT broker on which there are 2 topics: ‘coordinate’ & ‘ballstatus’ representing the target coordinate and the actual ball position respectively. Also connected to the MQTT server is a NodeJS server which handles receiving of user input (target coordinate), via a POST request, and sending the ballstatus via a WebSocket.

Data flows in 2 directions: User input -> NodeJS server -> MQTT broker -> PSoC (for the target coordinate) PSoC -> MQTT broker -> NodeJS server -> User interface (for the ballstatus)


Broker Interface
localhost:8088
publishes to: "ballstatus"
is subscribed to: "ballstatus"
Web Socket
PSoC 6
MQTT Broker
(mosquitto)
topics:[ballstatus, coordinate]
NodeJS Server (backend)
Frontend
is subscribed to: "coordinate"
publishes to: "coordinate"
POST request
Physical ball balancing table
Breadboard and jumper wires
Touchscreen
3 Servos

Methods

The PSoC measures the position of the ball using a four-wire resistive touchscreen using its ADC and GPIO pins. Because of how a four wire touchscreen works, the PSoC needs to reconfigure the GPIO pins at runtime, which was quite the challenge. Some filtering was used using custom circular buffers to stabilize the measurements. Next, a PID controller calculates the required angles for the 3 servo motors to correctly position the table top. Periodically the PSoC sends the measured and filtered ballstatus to the MQTT broker (currently at 200ms intervals). Whenever the PSoC receives a new payload of the “coordinate” topic, the PSoC updates the target values for x and y and the rest of the system adapts. These functions are split into FreeRTOS tasks which have their own priority and run interval timing.