1. OTAFU

Video See project folder:

https://drive.google.com/file/d/1yAR47tJvU8rI-BEMQbKNFhf3138G1v5b/view?usp=drive_link

2. GPIO Firmware Timing

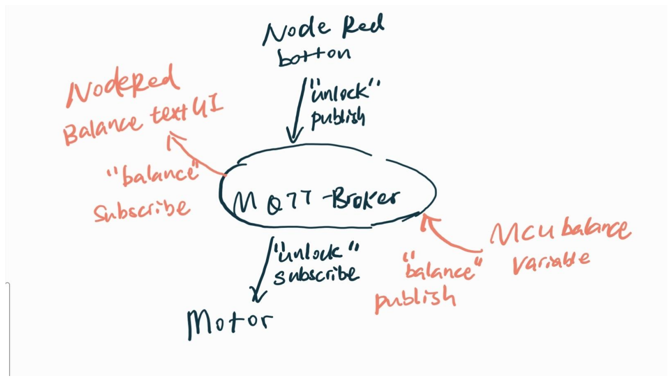
See project folder, avg time ~2.3ms

Video See project folder:

https://drive.google.com/file/d/1TpeKqyiOTSJPEPrCY00uKZxQXrZIkF69/view?usp=drive_link

3. Node-RED Design

- General program flow of your system. What communication does the device need to send to the cloud? What does the cloud do with this information? What does it send back?



Mainly the device will send the total amount of the money stored in the piggy bank to the cloud, and the cloud will show the count balance on the UI. The cloud will send back the remote button instruction for lock and unlock operation to the motor.

- List all the topics your system will use. Describe for each one the information that is sent on each topic (is it an integer? A Boolean? An array of integers? A JSON string?)

Topics

1. Unlock: User control on the cloud to press the unlock button on UI (send a boolean "true" to Mqtt), and the motor will receive the boolean "true" and implement unlocking/locking behavior.

2. Balance: MCU sends a message containing balance information (integer variable) to the cloud, and after login, it will display on the UI.

3. Clear Balance: User control on the cloud to press the reset button on UI (send a boolean “true” to Mqtt), and the MCU will clear the balance account, and the display on the piggy bank will clear the balance to zero.

- Describe for each topic in your system: Who generates the data? Who subscribes to each topic?

  Motor unlock: There should be a button on node-red to generate the lock/unlock message, and the motor should subscribe to the topic.

  Balance request: MCU will be the publisher to publish the Balance account variable and the node-red Cloud end will subscribe and show on the UI. Also, the cloud end can publish a clear request to MCU to clear the balance account.

- Present how you would divide your MCU Application code into threads, how the threads would communicate, etc.

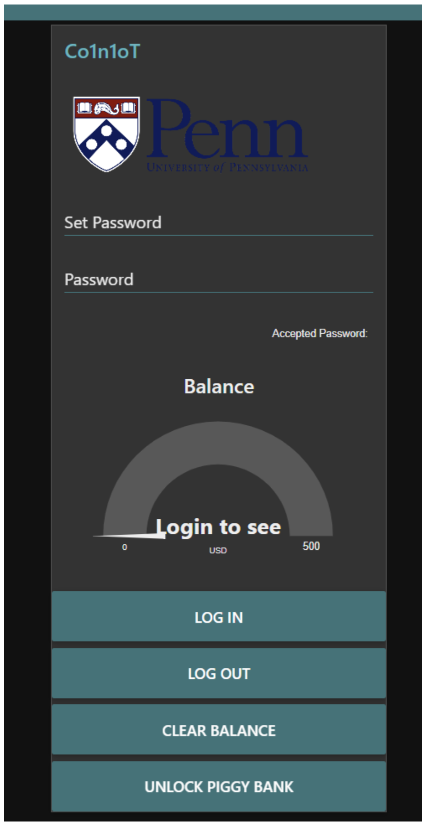
  Using FreeRTOS, there will be I2C to handle the light sensor, SPI to handle the LCD, and there are motor threads to perform the lock/unlock task, and there is a buzzer thread waiting for the I2C light sensor to set the handler to be true, and there is a motor thread waiting for the flag to be set from the cloud.

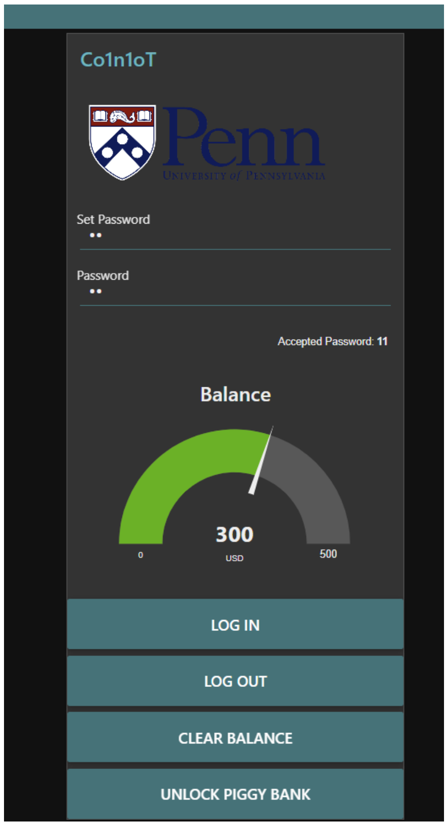
4. Node-RED Implementation

Node-Red:
http://172.190.188.203:1880
Node-Red UI:
http://172.190.188.203:1880/ui

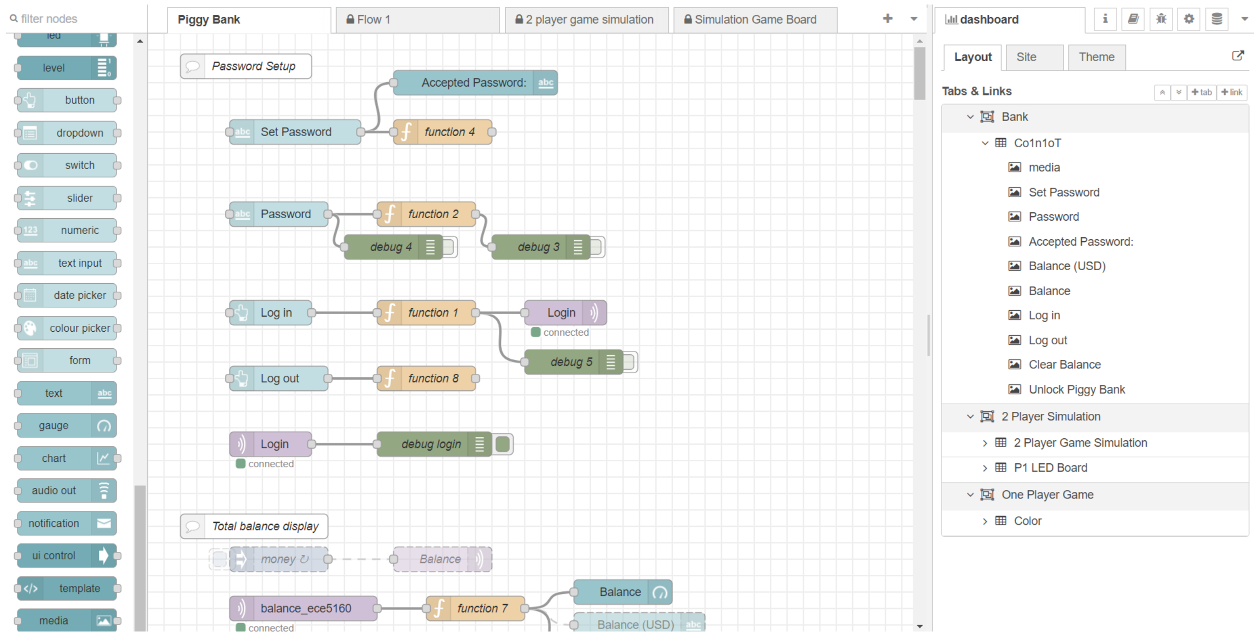
Video See project folder:
https://drive.google.com/file/d/11ruPOeUMdrW-T6uXNOtXhJf_wYz2ivWn/view?usp=drive_link

Frontend:





Backend:


filter nodes
led
level
button
dropdown
switch
slider
numeric
text input
date picker
colour picker
form
text
gauge
chart
audio out
notification
ui control
template
media
Piggy Bank
Flow 1
2 player game simulation
Simulation Game Board
Password Setup
Accepted Password:
Set Password
function 4
Password
function 2
debug 4
debug 3
Log in
function 1
Login
connected
debug 5
Log out
function 8
Login
connected
debug login
Total balance display
money
Balance
balance_ece5160
connected
function 7
Balance
Balance (USD)
dashboard
Layout
Site
Theme
Tabs & Links
Bank
Co1n1oT
media
Set Password
Password
Accepted Password:
Balance (USD)
Balance
Log in
Log out
Clear Balance
Unlock Piggy Bank
2 Player Simulation
2 Player Game Simulation
P1 LED Board
One Player Game
Color