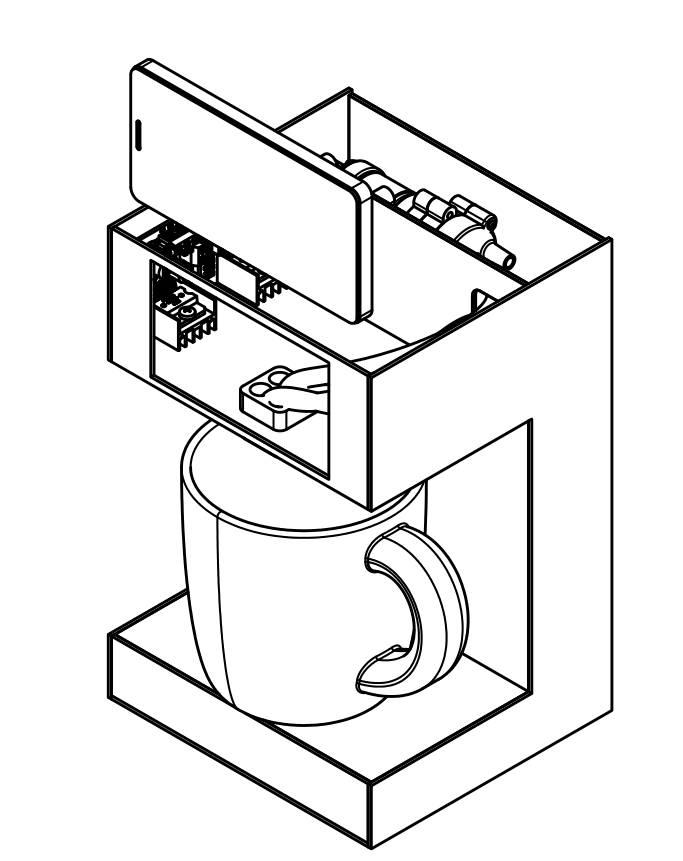
Beverage

Vending

Machine

With Cashless Payment System

|  |
| --- |
|  |
|  |
| Embedded system software 2018  Sitinut Waisara 59090030 Software Engineering International College KMITL |

C:\Users\max2-\Desktop\iclogoweb.png

### Introduction

In the present day, we are entering digital world. Everything is stored in digital world, including money. Cashless system become more convenient, secure and fast. It would be great if we can buy goods without human service and pay by cashless system.

### Objectives & Outcomes

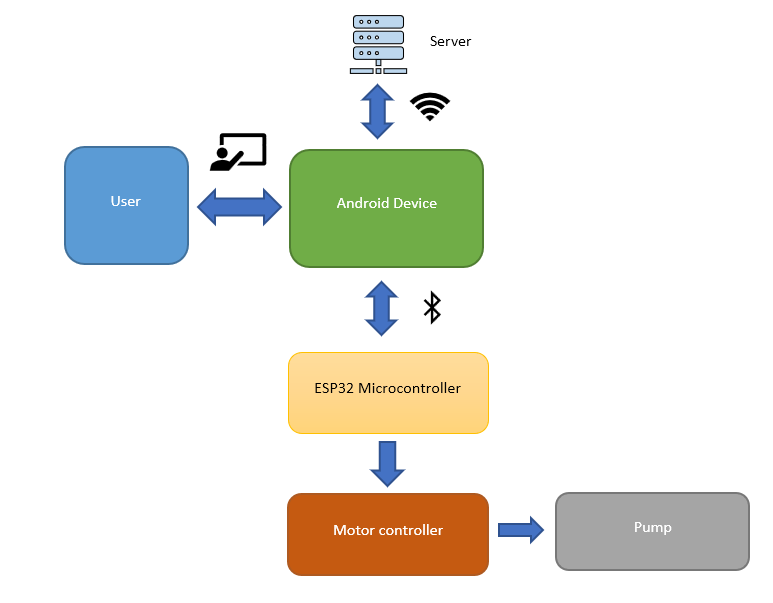
1. A working Beverage Vending Machine

* Controlled by Bluetooth.
* The machine can dispense each beverage individually.
* The machine can dispense liquid by volume precisely.

1. Simple and Easy user experience

* HMI with touchscreen via Android Device
* Simple and straightforward GUI
* Pay via QR Code (Thailand Standard)

### System Architecture



### Components

* 1. Android Device – Provides user interface (Screen, Touch Screen, Communication to server)
  2. ESP32 – Fast and cheap MCU + Bluetooth built-in
  3. Motor driver - L298N
  4. 12v Diaphragm pump
  5. Power supply, Regulator, etc.



Outlet nozzle

Inlet

Pumps

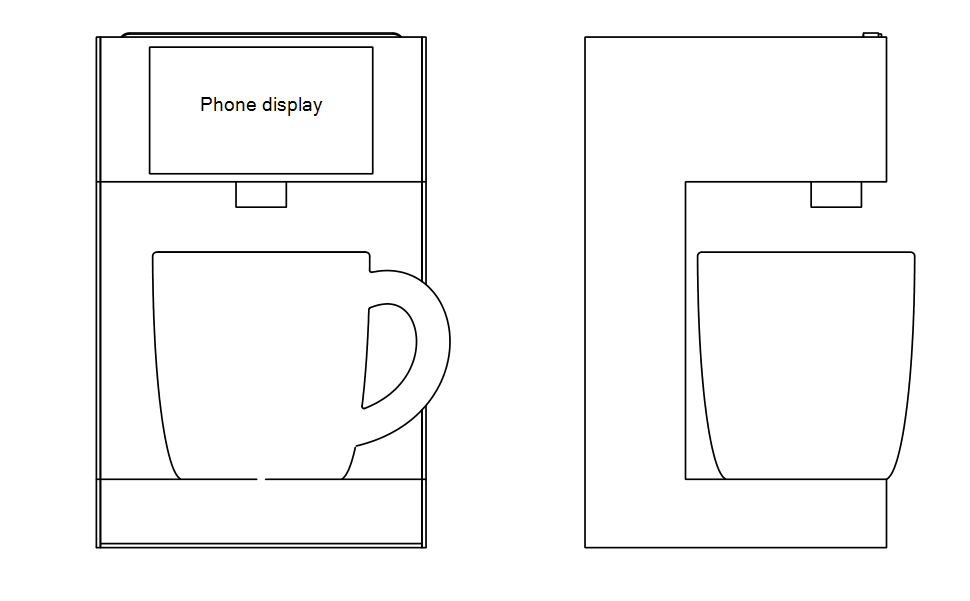
L298N Module

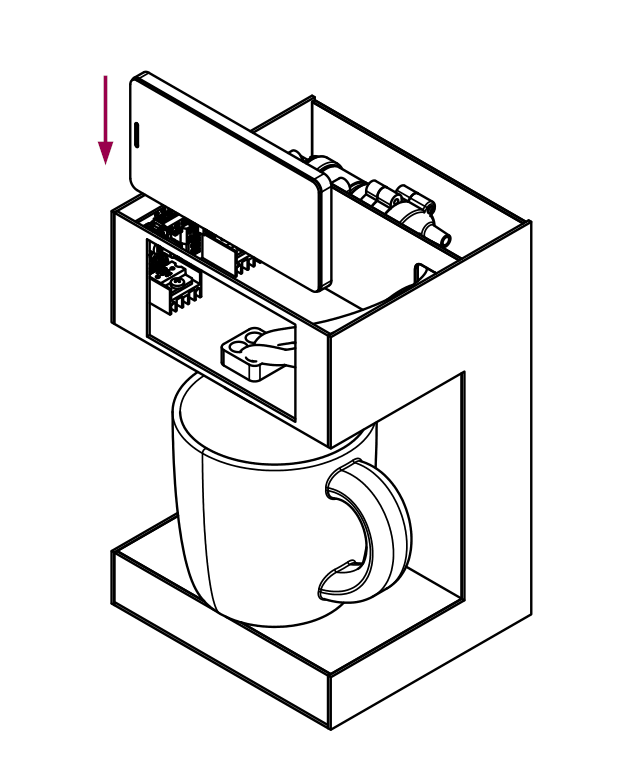
ESP32 MCU

Android Device

### Design

The Machine body were made from 2mm acrylic sheet which was cut by laser CNC machine. All parts were designed and assembled in CAD Program, this ensures that all parts will fit together.



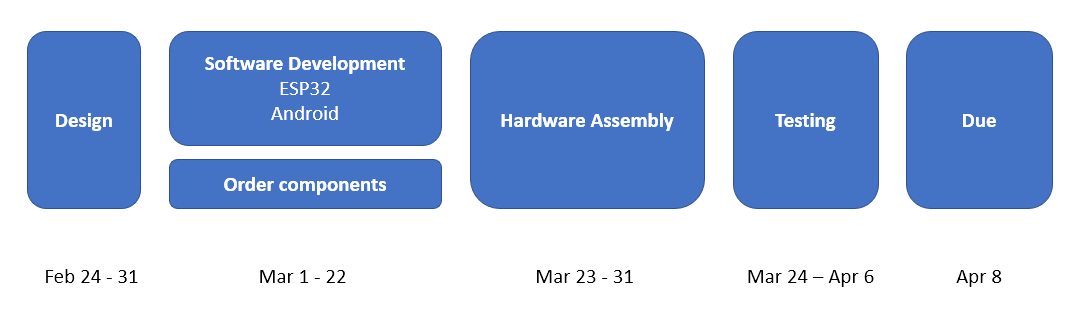


Phone can slide down easily

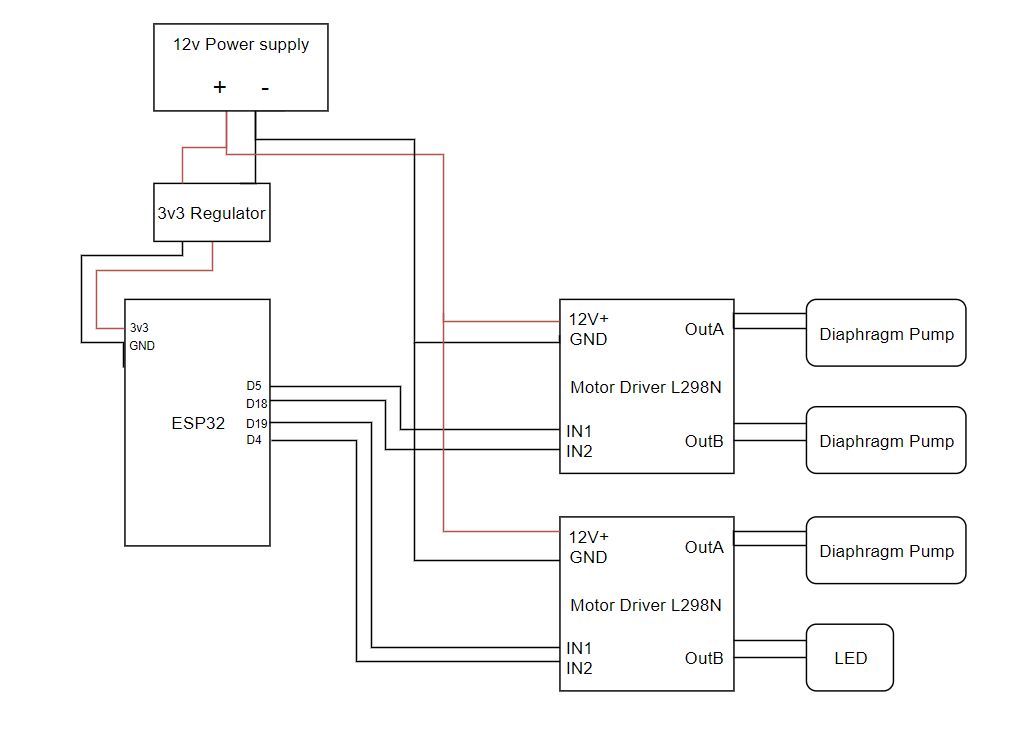
into the slot

## Implementation

### Timeline

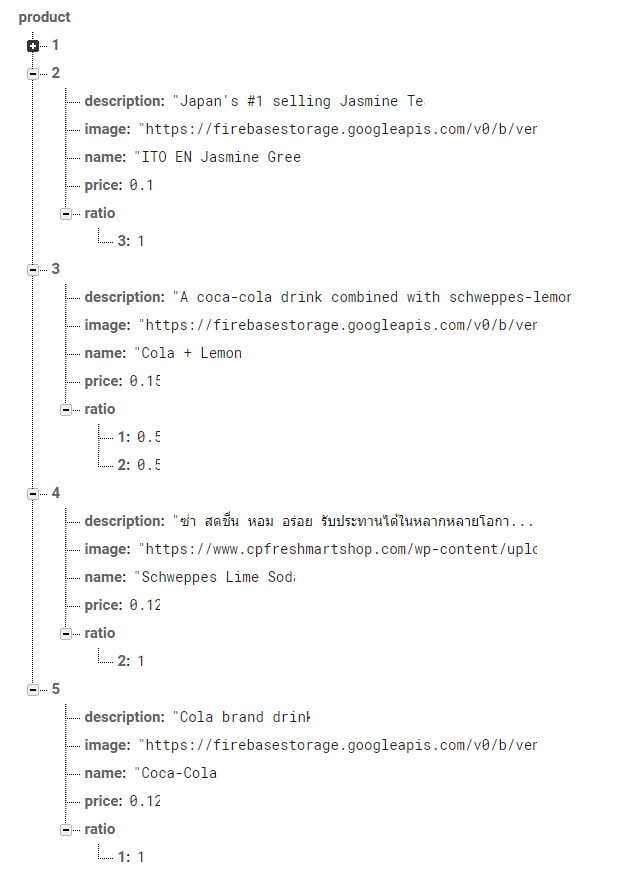


Circuit

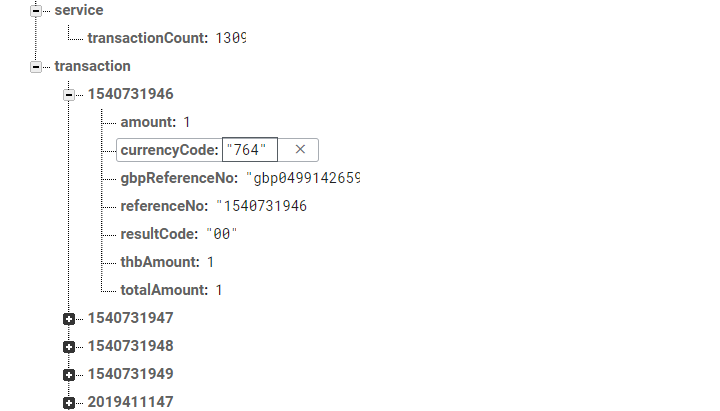


Database – Firebase Realtime Database

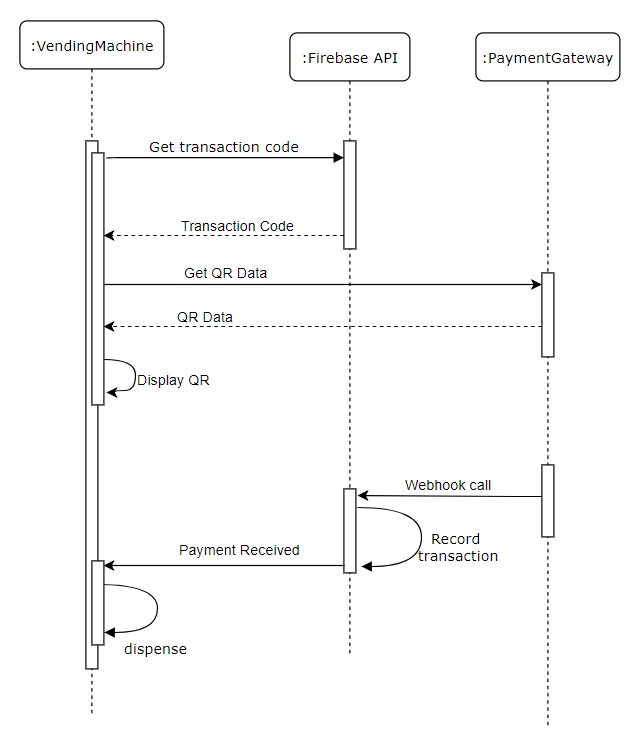
Data of all available drinks and information are stored in key-value structure.

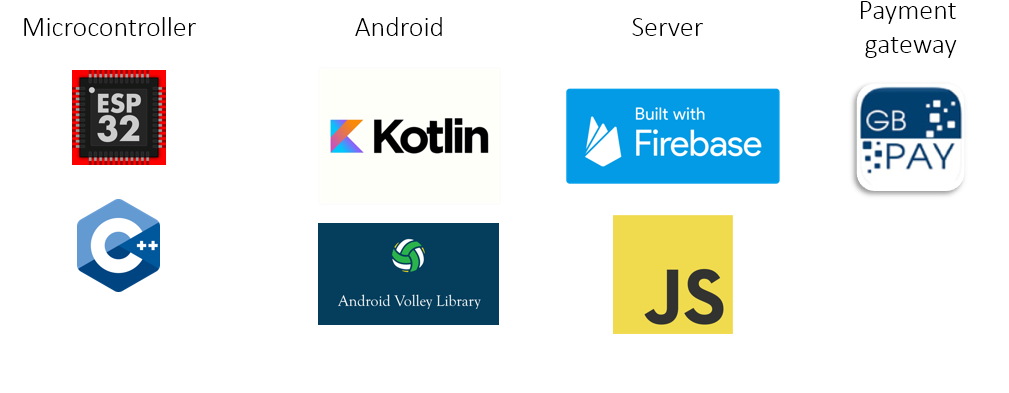


Transaction data



Sequence diagram of payment process



Tech Stack

Experiment and result

Flow rate can be obtained by equation below

= flow rate (ml/second)

= volume of dispensed liquid (milliliters)

= pump operating time (seconds)

For each dispenser, there’s some factor that affect flow rate such as power distribution, tube bending, viscosity of the ingredients. We try different motor power to minimize fluctuation of the flow rate.

|  |  |  |
| --- | --- | --- |
| Pump No. | Flow rate with 100% PWM duty cycle (ml/s) | Flow rate with 50% PWM duty cycle (ml/s) |
| 1 | 28.5 | 17.5 |
| 2 | 30 | 16.5 |
| 3 | 20 | 15 |
| Standard Deviation (SD) | 5.39 | 1.25 |

In conclusion, reducing pump power with 50% duty cycle helps the dispensed volume more accurate.

Finished product



Figure 3 Dispensing progress

Figure Main menu

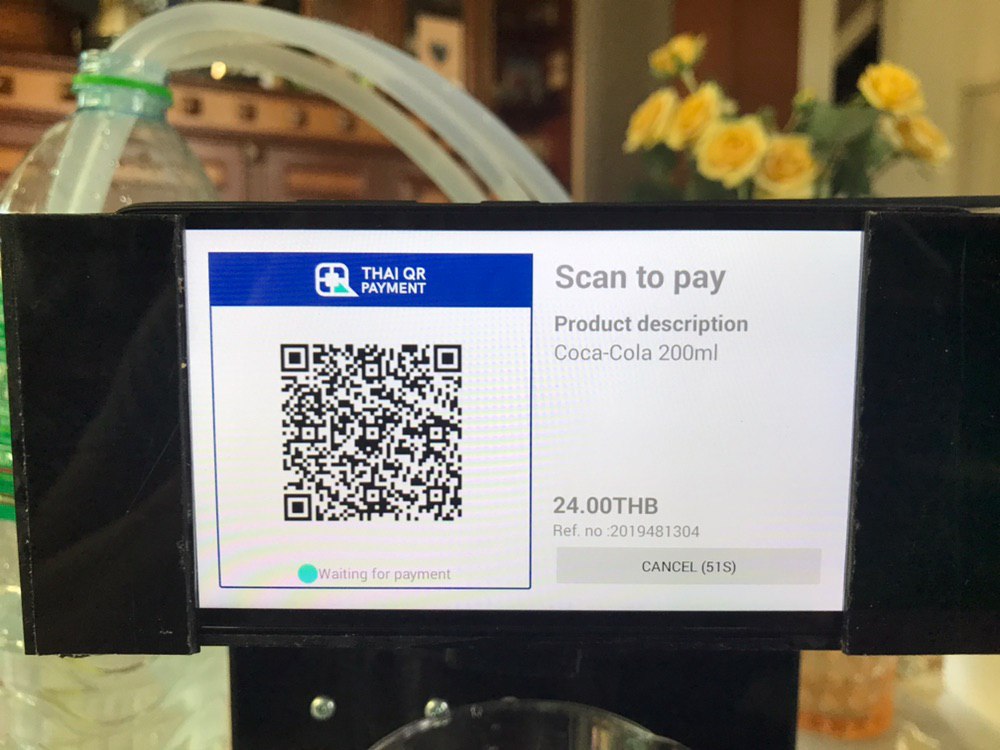


Figure Payment screen

Code on microcontroller

main.cpp

1. #include <Arduino.h>
2. #include "BluetoothSerial.h"
3. #include "VendingController.h"
4. #include "FastLED.h"
6. #if !defined(CONFIG\_BT\_ENABLED) || !defined(CONFIG\_BLUEDROID\_ENABLED)
7. #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
8. #endif
10. const uint8\_t led = 4;
11. const uint8\_t pump\_1 = 5;
12. const uint8\_t pump\_2 = 18;
13. const uint8\_t pump\_3 = 19;
15. bool isDispensing = false;
17. // setting PWM properties
18. const int freq = 5000;
19. const int ledChannel = 0;
20. const int resolution = 8;
22. VendingController vendingController(pump\_1,pump\_2,pump\_3);
23. BluetoothSerial SerialBT;




29. //Split string
30. String getValue(String data, char separator, int index)
31. {
32. int found = 0;
33. int strIndex[] = { 0, -1 };
34. int maxIndex = data.length() - 1;
36. for (int i = 0; i <= maxIndex && found <= index; i++) {
37. if (data.charAt(i) == separator || i == maxIndex) {
38. found++;
39. strIndex[0] = strIndex[1] + 1;
40. strIndex[1] = (i == maxIndex) ? i+1 : i;
41. }
42. }
43. return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
44. }
46. // Dispense the drink
47. void dispense(int ml1, int ml2, int ml3){
48. Serial.printf("Dispensing.. %d %d %d\n",ml1,ml2,ml3);
49. vendingController.dispense(ml1,ml2,ml3);
50. isDispensing = true;
51. }
53. void doAction(String in){
54. Serial.println("in = \'"+ in+"\'");
55. if(in == "chk"){
56. Serial.println("Send ok");
57. SerialBT.println("ok");
58. }
59. if (in.substring(0,3) == "dsp"){
60. int dspVal1 =  getValue(in,' ', 1).toInt();
61. int dspVal2 =  getValue(in,' ', 2).toInt();
62. int dspVal3 =  getValue(in,' ', 3).toInt();
63. for(int i = 1023; i >= 0; i--){
64. ledcWrite(ledChannel,i);
65. }
66. dispense(dspVal1,dspVal2,dspVal3);
67. }
68. }

71. void setup() {
72. Serial.begin(9600);
73. SerialBT.begin("ESP32 Vending Machine"); //Bluetooth device name
74. Serial.println("The device started, now you can pair it with bluetooth!");
75. // dispense(1250,833,500);
76. pinMode(led,OUTPUT);
77. // configure LED PWM functionalitites
78. ledcSetup(ledChannel, freq, resolution);
80. // attach the channel to the GPIO2 to be controlled
81. ledcAttachPin(led, ledChannel);
82. for(int i = 0; i < 1024; i++){
83. ledcWrite(ledChannel,i);
84. delay(10);
85. }

88. }
90. void ledFade(){
91. static int cur = 0;
92. static bool increase = true;
93. if (increase){
94. ledcWrite(ledChannel,cur);
95. cur +=10;
96. }
97. else{
98. ledcWrite(ledChannel,cur);
99. cur -=10;
100. }
101. if (cur >= 1024){
102. increase = false;
103. cur = 1024;
104. }
105. if (cur <= 200){
106. increase = true;
107. cur = 200;
108. }
110. }

113. void loop() {
114. if (Serial.available()) {
115. // SerialBT.write(Serial.read());
116. doAction(Serial.readString());
117. Serial.flush();
119. }
120. if (SerialBT.available()) {
121. String in =  SerialBT.readString();
122. in.replace("\n","");
123. Serial.println(in);
124. SerialBT.flush();
125. doAction(in);
126. }
128. vendingController.run();
130. if (isDispensing){
131. Serial.printf("Progress = %d\n", vendingController.getProgress());
132. SerialBT.println(vendingController.getProgress());
133. if (vendingController.getProgress()== 100){
134. isDispensing = false;
136. for(int i = 0; i < 5; i++){
137. vendingController.run();
138. ledcWrite(ledChannel,0);
139. delay(300);
140. ledcWrite(ledChannel,1024);
141. delay(300);
142. }
143. }
145. }
147. delay(50);

150. }

VendingController.h

1. #ifndef VendingController\_H
2. #define VendingController\_H
4. #include <Arduino.h>
6. class VendingController
7. {
9. public:
10. unsigned long int dispStopTime[3];
11. unsigned long int dispStartTime = 0;
12. const float flowRate = 25; // 1560 ml/minute -> 26 ml/sec
14. // setting PWM properties
15. const int freq = 30000;
16. const int resolution = 8;
17. int dutyCycle = 200;
19. uint8\_t disp\_pin[];
21. VendingController(uint8\_t disp1\_pin, uint8\_t disp2\_pin, uint8\_t disp3\_pin);
22. void dispense(int disp1\_ml, int disp2\_ml, int disp3\_ml);
23. void run();
24. int getProgress();
26. private:
27. int getDispenseTime(int ml);
28. };
30. #endif

VendingController.cpp

1. #include "VendingController.h"
2. #include "math.h"
4. //Init variable and setup PWM Pin
5. VendingController::VendingController(uint8\_t disp1\_pin, uint8\_t disp2\_pin, uint8\_t disp3\_pin)
6. {
7. disp\_pin[0] = disp1\_pin;
8. disp\_pin[1] = disp2\_pin;
9. disp\_pin[2] = disp3\_pin;
10. for(int i = 0; i < 3; i++){
11. pinMode(disp\_pin[i],OUTPUT);
12. ledcSetup(i+1, freq, resolution);       //Setup PWM Pin
13. ledcAttachPin(disp\_pin[i], i+1);      //Attach PWM Pin to PWM Channel
15. }
16. }
18. //Set the dispenser to dispense by volume
19. void VendingController::dispense(int disp1\_ml, int disp2\_ml, int disp3\_ml)
20. {
21. //The machine calculate end time for all dispenser
22. Serial.println("VendingCtrl : " + String(disp1\_ml, DEC) + " " + String(disp2\_ml, DEC) + " "+ String(disp3\_ml, DEC));
23. dispStopTime[0] = millis() + getDispenseTime(disp1\_ml);
24. dispStopTime[1] = millis() + getDispenseTime(disp2\_ml);
25. dispStopTime[2] = millis() + getDispenseTime(disp3\_ml);
26. dispStartTime = millis();
28. for(int i = 0; i < 3; i++){
29. Serial.println("VendingCtrl : " + String(dispStopTime[0], DEC) + " " + String(dispStopTime[1], DEC) + " "+ String(dispStopTime[2], DEC));
30. }
31. }
33. //Get the time that the dispenser need to turn on by volume
34. //Return: millis of dispense time
35. int VendingController::getDispenseTime(int ml)
36. {
37. return (ml / flowRate) \* 1000;
38. }
40. //Check periodically if the pump should run or stop
41. void VendingController::run()
42. {
43. static bool isOn[] = {false,false,false};
45. for (int i = 0; i < 3; i++)
46. {
47. //Not the stop time, then turn on pump
48. if (dispStopTime[i] > millis()){
49. if (!isOn[i]){
50. //digitalWrite(disp\_pin[i], HIGH);
51. ledcWrite(i+1 ,200);
52. isOn[i] = 1;
53. }
54. }
55. //Stop time passed, turn off pump.
56. else{
57. //digitalWrite(disp\_pin[i], LOW);
58. if (isOn[i]){
59. ledcWrite(i +1 ,0);
60. isOn[i] = 0;
61. }
62. }
63. }
64. }
66. //Calculate percent progress of dispensing progress
67. int VendingController::getProgress()
68. {
69. int min\_percent = INT\_MAX;
71. for (int i = 0; i < 3; i++){
73. if (dispStopTime[i] > millis())
74. {
75. min\_percent = min(100 - (int)((((float)dispStopTime[i] - millis()) / (dispStopTime[i] - dispStartTime)) \* 100), min\_percent);
76. }
77. else
78. {
79. min\_percent = min(min\_percent, 100);
80. }
81. }
83. return min\_percent;
84. }