

**TAP N' BREW: A SMART COFFEE MACHINE WITH COIN-OPERATED AND QR
CODE PAYMENT INTEGRATION**

CALAIS, GIULIANI

FABIAN, MEG ANGELINE

GALO, SHANLEY

MURILLO, PAMELA

SERRANO, KATE

BSIT – 3B

SUBMITTED TO:

MIKE BAE E. HAYAG

INTRODUCTION

Our project is entitled "Tap N' Brew: A Smart Coffee Machine with Coin-Operated and QR Code Payment Integration." It's a coffee machine but with a twist. It has both a coin slot and a touchscreen LCD display. Customers can use the touchscreen to choose between two types of coffee:

1. **Black Coffee.** Customers can customize this by adding sugar and creamer if they want.
2. **Brand Coffee.** This is a fixed recipe with no customization options. Customers can proceed directly to the next process after selecting it.

Also, after selecting their coffee, customers use the same touchscreen to choose their payment method. They can either pay through GCash by scanning a QR code or insert coins into the coin slot. Once the payment is complete, the machine automatically starts brewing the coffee.

Electronic devices can be programmed to perform tasks automatically with the Arduino. The ESP32-TFT LCD display works as a master and the Arduino Uno as a slave, or vice versa, for the commands to send to each other. Arduino is open source, which means that anyone can use it, change it, and share it. It helps our project to take payments and work smoothly, making it easier and safer for people to use.

Overall, this smart coffee machine is an innovation to the alternative of traditional vending coffee machines we see in our surroundings, such as in terminals, sari-sari stores, or in any other places. It is designed to provide a convenient way for people to enjoy a cup of coffee.

OBJECTIVES AND PURPOSE OF THE STUDY

With this project, the primary objective is to design and implement a smart coffee machine that accepts payments through coin-operated and QR code-based payment systems. Its purpose is to demonstrate the feasibility and the effectiveness of integrating QR code payments into a traditional coin-operated vending machine.

We expect to achieve the functionality of this project that will allow users to select and purchase a coffee product by either inserting coins or scanning a QR code to make a payment. The demonstration will show how the machine accepts payment and delivers the chosen coffee product and gives customers the easy way to pay. Additionally, the machine will also have features such as a user-friendly interface and secure coin and QR code payment processing, all of which will be showcased in the demonstration.

SIGNIFICANCE OF THE STUDY

The project, "Tap N' Brew: A Smart Coffee Machine with Coin-Operated and QR Code Payment Integration," is a significant innovation that addresses modern consumer needs for convenience and accessibility. It combines traditional coin-operated payment systems with advanced QR code payment technology, offering a seamless experience for coffee lovers. This machine is particularly valuable in today's cashless society, catering to individuals who prefer digital payment methods such as GCash.

The project is not only a reflection of the group's personal interest in leveraging technology for practical solutions but also a helpful tool for events and public spaces. It ensures that people, whether they have cash or prefer cashless transactions, can easily enjoy a cup of coffee. Its user-friendly interface and efficient payment options make it an ideal solution for venues like terminals, sari-sari stores, and other community spaces.

INSTRUCTIONS OF ARDUINO PROJECT.

1. Please read the printed instructions carefully before starting.
2. On the LCD screen, select your desired type of coffee.
3. If you select Black Coffee, you will be directed to the personalization page, where you can choose to add creamer or sugar.
 - If you select Kopiko Blanca, proceed to choose your payment method.
4. For payment:
 - If you choose GCash, you will be directed to the QR code page:
 - Scan the QR code to pay; the amount will depend on the selected coffee type.
 - If you choose the Coin Slot, you will be directed to the coin slot page where you can insert coins and view your balance.
5. After completing the payment, click "Next" to proceed to the cup detection page.
6. On the cup detection page, you will be prompted to place your cup. Ensure the cup is properly centered to proceed to brewing.
7. The brewing process will start; please wait until it is completed.
8. Once your coffee is ready, you may take your cup and enjoy your beverage.

COMPONENTS

Dependent

- Arduino uno
- Breadboard
- Logic Level Converter
- Centralized Power-supply
- ESP32-TFT LCD
- Universal Coin-slot
- Ultrasonic Sensor
- Servo motor (3x)
- Solenoid valve
- 4 Channel Relay Module
- Water Pump

Independent

- Water Dispenser Heater

CODES

```
1  #include <SoftwareSerial.h>
2  #include <Servo.h>
3
4  #define PULSE_PIN 2    // Pin connected to coin slot pulses
5  #define TRIGGER_PIN 10 // Ultrasonic sensor trigger pin
6  #define ECHO_PIN 9    // Ultrasonic sensor echo pin
7  #define BUZZER_PIN 8  // Buzzer pin
8  #define BUZZER_PIN2 7 // Buzzer pin 2
9  #define RELAY_PIN 5   // Solenoid pin
10 #define PUMP_PIN 6    // Water Pump pin
11 #define COFFEE_SERVO_PIN 11 // Pin connected to coffee servo
12 #define COFFEE2_SERVO_PIN 13 // Pin connected to creamer servo
13 #define SUGAR_SERVO_PIN 12 // Pin connected to sugar servo
14
15 volatile int pulseCount = 0;
16 const int cupDetectionThreshold = 5;
17
18 int balance = 0;
19 SoftwareSerial mySerial(3, 4);
20 Servo coffeeServo; // Servo for black coffee
21 Servo coffee2servo; // Servo for kopiko blanca
22 Servo sugarServo; // Servo for sugar
23
24 bool iscupPresent = false;
25
26 void pulseDetected() {
27     pulseCount++;
28 }
```

```
30 int determineCoinValue(int pulses) {
31     if (pulses == 1) return 1; // 1 Peso
32     if (pulses == 5) return 5; // 5 Pesos
33     if (pulses == 10) return 10; // 10 Pesos
34     if (pulses == 20) return 20; // 20 Pesos
35     return 0; // Unknown coin
36 }
37
38 bool isCupPresent() {
39     digitalWrite(TRIGGER_PIN, LOW);
40     delayMicroseconds(2);
41     digitalWrite(TRIGGER_PIN, HIGH);
42     delayMicroseconds(10);
43     digitalWrite(TRIGGER_PIN, LOW);
44
45     long duration = pulseIn(ECHO_PIN, HIGH);
46     int distance = duration * 0.034 / 2;
47     static int lastDistance = 0; // Store last distance
48 }
```

```

53 // Check if current reading is similar to last reading to debounce
54 if (abs(distance - lastDistance) <= 3) {
55     if (distance <= cupDetectionThreshold && !iscupPresent) {
56         // Serial.println("Cup detected!");
57         mySerial.println("CUP_PRESENT");
58         digitalWrite(BUZZER_PIN, HIGH);
59         delay(200);
60         digitalWrite(BUZZER_PIN, LOW);
61         iscupPresent = true;
62     } else if (distance > cupDetectionThreshold && iscupPresent) {
63         // Serial.println("Cup removed!");
64         mySerial.println("CUP_REMOVE");
65         tone(BUZZER_PIN, 1000, 500);
66         iscupPresent = false;
67     }
68 }
69
70 lastDistance = distance; // Update last distance
71 }

```

```

74 void handleBrewCommand(String command) {
75     if (command == "START_BREW_BLACK_COFFEE_YES") {
76         Serial.println("Brewing Black Coffee with Creamer and Sugar...");
77         dispenseSugar();
78         dispenseCoffee(); // Assumed to dispense coffee with sugar and cream
79     } else if (command == "START_BREW_BLACK_COFFEE_NO") {
80         Serial.println("Brewing Plain Black Coffee...");
81         dispenseCoffee(); // Only dispense coffee powder
82     } else if (command == "START_BREW_KOPIKO") {
83         Serial.println("Brewing Kopiko Blanca...");
84         dispenseCoffee2(); // Dispense pre-mixed Kopiko Blanca
85     }
86
87     // Wait until all dispensing actions are completed
88     digitalWrite(RELAY_PIN, HIGH); // Ensure the valve is closed after operations
89     delay(1000); // Wait a bit to ensure all actions are settled
90
91     // Notify completion of brewing
92     // Serial.println("Brewing complete.");
93     mySerial.println("BREW_DONE");
94
95
96     playSuccessTone(); // Play a success tone when brewing is complete
97 }
98
99 void playSuccessTone() {
100
101     delay(2000);
102     tone(BUZZER_PIN2, 2000, 1000); // Play a higher tone (2000 Hz) for 1 second
103 }
104

```

```

106 void dispenseCoffee() {
107     coffee2servo.detach();
108     sugarServo.detach();
109
110     coffeeServo.attach(COFFEE_SERVO_PIN);
111
112     // Dispense the coffee
113     for (int i = 0; i < 2; i++) {
114         coffeeServo.write(116); // Dispense position
115         delay(2000);
116         coffeeServo.write(0); // Return to initial position
117         delay(2000);
118     }
119     coffeeServo.detach(); // Detach the servo after operation
120
121     // Manage pump and relay
122     int durations[] = {3000, 1000}; // durations for the pump and relay
123     for (int i = 0; i < 2; i++) {
124         digitalWrite(RELAY_PIN, LOW);
125         digitalWrite(PUMP_PIN, LOW);
126
127         delay(durations[i]);
128         digitalWrite(PUMP_PIN, HIGH);
129         digitalWrite(RELAY_PIN, HIGH);
130         // Serial.println("Pump and relay stopped.");
131         delay(1000);
132     }
133 }

```

```

141 void dispenseCoffee2() {
142     coffeeServo.detach();
143     sugarServo.detach();
144
145     coffee2servo.attach(COFFEE2_SERVO_PIN);
146
147     // Start servo movement for dispensing
148     Serial.println("Starting servo movement for dispensing...");
149     for (int i = 0; i < 20; i++) {
150         coffee2servo.write(60);
151         delay(1000); // Dispense position
152         coffee2servo.write(0);
153         delay(1000); // Return to initial position
154     }
155
156     coffee2servo.detach(); // Detach the servo to save power
157     // Serial.println("Servo movement complete. Servo detached.");
158
159     // Array of durations for pump and valve operation
160     int durations[] = {3000, 1000}; // First for 3 seconds, then 1 seconds
161 }

```

```

162 // Open the valve and start the pump based on durations
163 for (int i = 0; i < 2; i++) {
164     digitalWrite(RELAY_PIN, LOW);
165     digitalWrite(PUMP_PIN, LOW);
166
167     delay(durations[i]); // Run for the specified duration
168     digitalWrite(PUMP_PIN, HIGH);
169     digitalWrite(RELAY_PIN, HIGH);
170
171     delay(1000);
172 }
173
174 // Serial.println("Pump and valve operation complete.");
175 }

```

```

185
186 void dispenseSugar() {
187     coffeeServo.detach();
188     coffee2servo.detach();
189
190     sugarServo.attach(SUGAR_SERVO_PIN);
191
192     // Start the servo to dispense sugar
193     for (int i = 0; i < 5; i++) {
194         sugarServo.write(48);
195         delay(1000);
196         sugarServo.write(0);
197         delay(1000);
198     }
199     sugarServo.detach(); // Detach the servo
200
201 }

```

```

205 void setup() {
206     Serial.begin(9600);
207     mySerial.begin(4800);
208
209     pinMode(RELAY_PIN, OUTPUT);
210     digitalWrite(RELAY_PIN, HIGH);
211     pinMode(PUMP_PIN, OUTPUT);
212     digitalWrite(PUMP_PIN, HIGH);
213     pinMode(PULSE_PIN, INPUT_PULLUP);
214     attachInterrupt(digitalPinToInterrupt(PULSE_PIN), pulseDetected, FALLING);
215
216     coffeeServo.write(0); |
217     coffee2servo.write(0);
218     sugarServo.write(0);
219
220
221     pinMode(TRIGGER_PIN, OUTPUT);
222     pinMode(ECHO_PIN, INPUT);
223     pinMode(BUZZER_PIN, OUTPUT);
224
225     pulseCount = 0;
226 }

```

```

228 void loop() {
229     if (pulseCount > 0) {
230         delay(100);
231         int coinValue = determineCoinValue(pulseCount);
232
233         if (coinValue > 0) {
234             balance += coinValue;
235             mySerial.println(balance);
236             // Serial.println("Sent last balance to ESP32: " + String(balance));
237         }
238         pulseCount = 0;
239     }
240
241     if (mySerial.available()) {
242         String command = mySerial.readStringUntil('\n');
243         command.trim();
244
245
246         Serial.println("Received command: " + command); // Debug: log received command
247     }

```

```

246 Serial.println("Received command: " + command); // Debug: log received command
247
248 if (command == "CHECK_BALANCE") {
249     mySerial.println(balance);
250     Serial.println("Sent last balance to ESP32: " + String(balance));
251 } else if (command == "RESET_BALANCE") {
252     balance = 0;
253     Serial.println("Balance reset.");
254 } else if (command == "RESET_ALL") {
255     balance = 0; // Reset the balance
256     pulseCount = 0; // Reset the pulse count
257     isCupPresent = false; // Reset cup detection flag
258
259     // Reset all servos to their initial positions
260     coffeeServo.attach(COFFEE_SERVO_PIN);
261     coffeeServo.write(0);
262     coffeeServo.detach();

```

```

263
264     coffee2servo.attach(COFFEE2_SERVO_PIN);
265     coffee2servo.write(0);
266     coffee2servo.detach();
267
268     sugarServo.attach(SUGAR_SERVO_PIN);
269     sugarServo.write(0);
270     sugarServo.detach();
271
272     // Turn off any active components
273     digitalWrite(BUZZER_PIN, LOW);
274     digitalWrite(BUZZER_PIN2, LOW);
275     digitalWrite(RELAY_PIN, HIGH);
276     digitalWrite(PUMP_PIN, HIGH);
277
278     // Serial.println("Resetting all components...");

```

```

278     // Serial.println("Resetting all components...");
279 }
280
281 if (command == "CHECK_CUP") {
282     isCupPresent();
283 } else if (command.startsWith("START_BREW"))
284 {
285     handleBrewCommand(command);
286 }
287 }
288 }

```

BLUEPRINT OF ARUINO PROJECT

