

Penny Pal Codes

pins on XIAOESP32C6

```
BUTTON_1  D2  
BUTTON_2  D3  
BUTTON_5  D1  
BUTTON_10 D5  
  
NEOPIXEL  D4  
  
IR  SENSOR D8  
  
SOLENOID 1 D9  
SOLENOID 2 D10
```

JQ6500 Audio

```
#include <Arduino.h>  
#include <JQ6500_Serial.h>
```

```

// XIAO ESP32-C6: D6 = GPIO17 (TX), D7 = GPIO18 (RX)
#define MP3_RX_PIN 17 // D7 → connect to JQ6500 TX
#define MP3_TX_PIN 16 // D6 → connect to JQ6500 RX

HardwareSerial mp3Serial(1); // UART1
JQ6500_Serial mp3(mp3Serial);

void setup() {
  Serial.begin(115200);
  delay(500);
  Serial.println("Ready. Enter command:");

  // Begin UART1 with D6/D7 pins
  mp3Serial.begin(9600, SERIAL_8N1, MP3_RX_PIN, MP3_TX_PIN);
  delay(3000); // JQ6500 boot time

  mp3.reset();
  delay(1000);

  mp3.setSource(MP3_SRC_BUILTIN);
  delay(500);

  mp3.setVolume(30); // MAX volume
  delay(200);
}

void loop() {
  if (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    input.trim();

    if (input == "e") {
      Serial.println("Playing 0001.mp3");
      mp3.playFileByIndexNumber(1);
    }
    else if (input == "1" || input == "2" || input == "5" ||

```

```

input == "10") {
    Serial.println("Playing 0002.mp3");
    mp3.playFileByIndexNumber(2);
}
else if (input == "h") {
    Serial.println("Playing 0003.mp3");
    mp3.playFileByIndexNumber(3);
}
else if (input == "r") {
    Serial.println("Playing 0004.mp3");
    mp3.playFileByIndexNumber(4);
}
else {
    Serial.println("Invalid input");
}
}
}

```

BUTTON INPUT WITH DENOMINATION

```

#define BUTTON_1  D2
#define BUTTON_2  D3
#define BUTTON_5  D1
#define BUTTON_10 D5

bool lastB1 = LOW;
bool lastB2 = LOW;
bool lastB5 = LOW;
bool lastB10 = LOW;

void setup() {
    Serial.begin(115200);
}

```

```

pinMode(BUTTON_1, INPUT);
pinMode(BUTTON_2, INPUT);
pinMode(BUTTON_5, INPUT);
pinMode(BUTTON_10, INPUT);
}

void loop() {

    bool b1 = digitalRead(BUTTON_1);
    bool b2 = digitalRead(BUTTON_2);
    bool b5 = digitalRead(BUTTON_5);
    bool b10 = digitalRead(BUTTON_10);

    if (b1 == HIGH && lastB1 == LOW) {
        Serial.println("BUTTON 1 PRESSED");
        delay(50);
    }

    if (b2 == HIGH && lastB2 == LOW) {
        Serial.println("BUTTON 2 PRESSED");
        delay(50);
    }

    if (b5 == HIGH && lastB5 == LOW) {
        Serial.println("BUTTON 5 PRESSED");
        delay(50);
    }

    if (b10 == HIGH && lastB10 == LOW) {
        Serial.println("BUTTON 10 PRESSED");
        delay(50);
    }

    lastB1 = b1;
    lastB2 = b2;
    lastB5 = b5;
}

```

```
    lastB10 = b10;
}
```

SOLENOID TEST (2)

```
#define SOLENOID D10
// the setup function runs once when you press reset or power
the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(SOLENOID, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(SOLENOID, HIGH); // change state of the LED b
y setting the pin to the HIGH voltage level
    delay(2000);                  // wait for a second
    digitalWrite(SOLENOID, LOW);  // change state of the LED b
y setting the pin to the LOW voltage level
    delay(2000);                  // wait for a second
}
```

SOLENOID LOCKS AND UNLOCKS WITH BUTTON INPUT

```

#define BUTTON_1    D2
#define BUTTON_2    D3
#define BUTTON_5    D1
#define BUTTON_10   D5

#define SOLENOID     D10

void setup() {
  Serial.begin(115200);

  pinMode(BUTTON_1, INPUT_PULLUP);
  pinMode(BUTTON_2, INPUT_PULLUP);
  pinMode(BUTTON_5, INPUT_PULLUP);
  pinMode(BUTTON_10, INPUT_PULLUP);

  pinMode(SOLENOID, OUTPUT);
  digitalWrite(SOLENOID, LOW);
}

void pulseSolenoid() {
  digitalWrite(SOLENOID, HIGH);
  delay(200);
  digitalWrite(SOLENOID, LOW);
}

void loop() {

  if (digitalRead(BUTTON_1) == LOW) {
    Serial.println("1");
    pulseSolenoid();
    while (digitalRead(BUTTON_1) == LOW);
    delay(50);
  }

  if (digitalRead(BUTTON_2) == LOW) {

```

```

    Serial.println("2");
    pulseSolenoid();
    while (digitalRead(BUTTON_2) == LOW);
    delay(50);
}

if (digitalRead(BUTTON_5) == LOW) {
    Serial.println("5");
    pulseSolenoid();
    while (digitalRead(BUTTON_5) == LOW);
    delay(50);
}

if (digitalRead(BUTTON_10) == LOW) {
    Serial.println("10");
    pulseSolenoid();
    while (digitalRead(BUTTON_10) == LOW);
    delay(50);
}
}

```

ADDED RUNNING TOTAL WITH DELAY FOR SOLENOID

- Prints the coin value pressed.
- Keeps a running total.
- Prints the updated total after every coin.
- Activates the solenoid after a short delay.
- Triggers only once per button press.

```

#define BUTTON_1    D2
#define BUTTON_2    D3
#define BUTTON_5    D1
#define BUTTON_10   D5

#define SOLENOID     D10

int total = 0;

void setup() {
  Serial.begin(115200);

  pinMode(BUTTON_1, INPUT_PULLUP);
  pinMode(BUTTON_2, INPUT_PULLUP);
  pinMode(BUTTON_5, INPUT_PULLUP);
  pinMode(BUTTON_10, INPUT_PULLUP);

  pinMode(SOLENOID, OUTPUT);
  digitalWrite(SOLENOID, LOW);

  Serial.println("Coin Counter Ready");
}

void pulseSolenoid() {
  digitalWrite(SOLENOID, HIGH);
  delay(600);
  digitalWrite(SOLENOID, LOW);
}

void addCoin(int value) {
  Serial.print("Coin Inserted: ");
  Serial.println(value);

  total += value;
}

```

```

Serial.print("Total: ");
Serial.println(total);

delay(600); // small delay before solenoid movement

pulseSolenoid();
}

void loop() {

  if (digitalRead(BUTTON_1) == LOW) {
    addCoin(1);
    while (digitalRead(BUTTON_1) == LOW);
    delay(50);
  }

  if (digitalRead(BUTTON_2) == LOW) {
    addCoin(2);
    while (digitalRead(BUTTON_2) == LOW);
    delay(50);
  }

  if (digitalRead(BUTTON_5) == LOW) {
    addCoin(5);
    while (digitalRead(BUTTON_5) == LOW);
    delay(50);
  }

  if (digitalRead(BUTTON_10) == LOW) {
    addCoin(10);
    while (digitalRead(BUTTON_10) == LOW);
    delay(50);
  }
}

```

GOAL + BUTTON INPUT + SOLENOID + JQ6500

- Goal = **50**
- Every button press:
 - Print coin value
 - Play coin sound (**track 2**)
 - Wait 500 ms
 - Fire solenoid
 - Add to total
 - Print total
- At **25% of goal (12.5 → 13 and above)**:
 - Play **track 3** once
- At **goal reached (50 and above)**:
 - Play **track 4** once

Track mapping:

- 0002.mp3 = coin inserted
- 0003.mp3 = 25% milestone
- 0004.mp3 = goal achieved

```
#include <Arduino.h>
#include <JQ6500_Serial.h>

#define BUTTON_1    D2
#define BUTTON_2    D3
#define BUTTON_5    D1
#define BUTTON_10   D5
```

```

#define SOLENOID    D10

#define MP3_RX_PIN 17    // D7 -> JQ6500 TX
#define MP3_TX_PIN 16    // D6 -> JQ6500 RX

HardwareSerial mp3Serial(1);
JQ6500_Serial mp3(mp3Serial);

const int GOAL = 50;
int total = 0;

bool quarterPlayed = false;
bool goalPlayed = false;

void pulseSolenoid() {
    digitalWrite(SOLENOID, HIGH);
    delay(100);
    digitalWrite(SOLENOID, LOW);
}

void checkMilestones() {

    if (!quarterPlayed && total >= (GOAL / 4)) {
        quarterPlayed = true;

        Serial.println("25% Goal Reached!");
        mp3.playFileByIndexNumber(3);
    }

    if (!goalPlayed && total >= GOAL) {
        goalPlayed = true;

        Serial.println("GOAL ACHIEVED!");
        mp3.playFileByIndexNumber(4);
    }
}

```

```

}

void addCoin(int value) {

    Serial.print("Coin Inserted: ");
    Serial.println(value);

    // Play coin sound immediately
    mp3.playFileByIndexNumber(2);

    // Wait before unlocking
    delay(500);

    pulseSolenoid();

    total += value;

    Serial.print("Total: ");
    Serial.print(total);
    Serial.print(" / ");
    Serial.println(GOAL);

    checkMilestones();
}

void setup() {

    Serial.begin(115200);

    pinMode(BUTTON_1, INPUT_PULLUP);
    pinMode(BUTTON_2, INPUT_PULLUP);
    pinMode(BUTTON_5, INPUT_PULLUP);
    pinMode(BUTTON_10, INPUT_PULLUP);

    pinMode(SOLENOID, OUTPUT);
    digitalWrite(SOLENOID, LOW);

```

```

    delay(500);

    mp3Serial.begin(9600, SERIAL_8N1, MP3_RX_PIN, MP3_TX_PIN);

    delay(3000);

    mp3.reset();
    delay(1000);

    mp3.setSource(MP3_SRC_BUILTIN);
    delay(500);

    mp3.setVolume(30);

    Serial.println("-----");
    Serial.print("Goal: ");
    Serial.println(GOAL);
    Serial.println("Ready");
    Serial.println("-----");
}

void loop() {

    if (digitalRead(BUTTON_1) == LOW) {
        addCoin(1);
        while (digitalRead(BUTTON_1) == LOW);
        delay(50);
    }

    if (digitalRead(BUTTON_2) == LOW) {
        addCoin(2);
        while (digitalRead(BUTTON_2) == LOW);
        delay(50);
    }
}

```

```

if (digitalRead(BUTTON_5) == LOW) {
    addCoin(5);
    while (digitalRead(BUTTON_5) == LOW);
    delay(50);
}

if (digitalRead(BUTTON_10) == LOW) {
    addCoin(10);
    while (digitalRead(BUTTON_10) == LOW);
    delay(50);
}
}

```

NEOPIXEL

```

#include <Adafruit_NeoPixel.h>

#define PIXEL_PIN    D4
#define NUM_PIXELS   4

Adafruit_NeoPixel pixels(NUM_PIXELS, PIXEL_PIN, NEO_GRB + NEO_KHZ800);

void setup() {
    pixels.begin();
    pixels.clear();
    pixels.show();
}

void loop() {

    // Chase white

```

```
for (int i = 0; i < NUM_PIXELS; i++) {  
    pixels.clear();  
    pixels.setPixelColor(i, pixels.Color(50, 50, 50));  
    pixels.show();  
    delay(300);  
}  
  
// All Red  
pixels.fill(pixels.Color(50, 0, 0));  
pixels.show();  
delay(1000);  
  
// All Green  
pixels.fill(pixels.Color(0, 50, 0));  
pixels.show();  
delay(1000);  
  
// All Blue  
pixels.fill(pixels.Color(0, 0, 50));  
pixels.show();  
delay(1000);  
  
// All Off  
pixels.clear();  
pixels.show();  
delay(1000);  
}
```

OTA READY

```

#include <WiFi.h>
#include <ArduinoOTA.h>

const char* ssid = "MiCho";
const char* password = "Hello1234";

void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println();
  Serial.println("Starting...");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println();
  Serial.println("WiFi Connected");

  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());

  ArduinoOTA.setHostname("PennyPal");

  ArduinoOTA.onStart([]() {
    Serial.println("OTA Update Started");
  });
}

```

```

ArduinoOTA.onEnd([]() {
    Serial.println("\nOTA Update Complete");
});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress * 100) / total);
});

ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]\n", error);

    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
});

ArduinoOTA.begin();

Serial.println("OTA Ready");
Serial.println("Hostname: PennyPal");
}

void loop() {
    ArduinoOTA.handle();
}

```

WiFi Connected!

IP Address: 10.107.239.122

Starting...

Connecting to WiFi.....

WiFi Connected

IP Address: 10.107.239.122

OTA Ready

Hostname: PennyPal

OTA TEST

```
#include <WiFi.h>
#include <ArduinoOTA.h>

const char* ssid = "MiCho";
const char* password = "Hello1234";

void setup() {
  Serial.begin(115200);
  delay(1000);

  Serial.println("\nBooting...");

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi Connected");
```

```

Serial.print("IP Address: ");
Serial.println(WiFi.localIP());

ArduinoOTA.setHostname("PennyPal");

ArduinoOTA.onStart([]() {
    Serial.println("OTA Start");
});

ArduinoOTA.onEnd([]() {
    Serial.println("\nOTA Complete");
});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress * 100) / total);
});

ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]\n", error);
});

ArduinoOTA.begin();

Serial.println("OTA Ready");
}

void loop() {
    ArduinoOTA.handle();

    static unsigned long last = 0;

    if (millis() - last > 3000) {
        Serial.println("OTA TEST V1");
        last = millis();
    }
}

```

```
}  
}
```

WiFi Connected

IP Address: 10.107.239.122

OTA Ready

OTA TEST V1

OTA TEST V1

OTA TEST V1

OTA TEST V1

Setting OTA Password

```
#include <WiFi.h>  
#include <ArduinoOTA.h>  
  
const char* ssid = "MiCho";  
const char* password = "Hello1234";  
  
void setup() {  
  Serial.begin(115200);  
  delay(1000);  
  
  Serial.println("\nBooting...");  
  
  WiFi.mode(WIFI_STA);  
  WiFi.begin(ssid, password);  
  
  while (WiFi.status() != WL_CONNECTED) {
```

```

    delay(500);
    Serial.print(".");
}

Serial.println("\nWiFi Connected");
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());

ArduinoOTA.setHostname("PennyPal");

// OTA password
ArduinoOTA.setPassword("1234");

ArduinoOTA.onStart([]() {
    Serial.println("OTA Start");
});

ArduinoOTA.onEnd([]() {
    Serial.println("\nOTA Complete");
});

ArduinoOTA.onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress * 100) / total);
});

ArduinoOTA.onError([](ota_error_t error) {
    Serial.printf("Error[%u]\n", error);
});

ArduinoOTA.begin();

Serial.println("OTA Ready");
Serial.println("OTA Password: 1234");
}

```

```

void loop() {
  ArduinoOTA.handle();

  static unsigned long last = 0;

  if (millis() - last > 3000) {
    Serial.println("OTA TEST V1");
    last = millis();
  }
}

```

CONNECTED TO KODULAR COMPANION APP VIA FIREBASE

```

// =====
// PENNY PAL
// XIAO ESP32-C6
//
// D1 = ₹1
// D2 = ₹2
// D5 = ₹5
// D3 = ₹10
//
// Parent App:
//   writes /piggy/goal
//
// ESP32:
//   reads /piggy/goal
//   writes /piggy/total
//   writes /piggy/percentage
//   writes /piggy/lastCoin

```

```

//
// New Goal:
//   resets savings challenge
//
// Reboot:
//   restores progress from Firebase
// =====

// ----- WIFI -----

#include <WiFi.h>

// ----- FIREBASE -----

#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

// ----- WIFI CREDENTIALS -----

const char* ssid = "MiCho";
const char* password = "Hello1234";

// ----- FIREBASE CONFIG -----

#define API_KEY "AIzaSyDpefz07K0uYyj0mZ1YtvJkm5g3z5HU0-I"
#define DATABASE_URL "https://smart-piggy-effc6-default-rtdb.
asia-southeast1.firebaseio.com/"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// =====
// BUTTON PINS
// =====

```

```

#define BTN_1    D1    // ₹1
#define BTN_2    D2    // ₹2
#define BTN_5    D5    // ₹5
#define BTN_10   D3    // ₹10

// =====
// SAVINGS DATA
// =====

int totalAmount = 0;
int goal = 0;
int lastGoal = -1;

// =====
// TIMERS
// =====

unsigned long lastFirebaseRead = 0;
const unsigned long firebaseInterval = 2000;

// =====
// BUTTON STATE MEMORY
//
// INPUT_PULLUP means:
//
// HIGH = not pressed
// LOW  = pressed
// =====

bool lastBtn1 = HIGH;
bool lastBtn2 = HIGH;
bool lastBtn5 = HIGH;
bool lastBtn10 = HIGH;

// =====

```

```

// SETUP
// =====

void setup() {

    Serial.begin(115200);

    Serial.println();
    Serial.println("=====");
    Serial.println("PENNY PAL STARTING");
    Serial.println("=====");

    // Internal pullups

    pinMode(BTN_1, INPUT_PULLUP);
    pinMode(BTN_2, INPUT_PULLUP);
    pinMode(BTN_5, INPUT_PULLUP);
    pinMode(BTN_10, INPUT_PULLUP);

    // ----- WIFI -----

    Serial.print("Connecting WiFi");

    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.println("WiFi Connected");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());

    // ----- FIREBASE -----

```

```

config.api_key = API_KEY;
config.database_url = DATABASE_URL;
config.token_status_callback = tokenStatusCallback;

Firebase.signUp(&config, &auth, "", "");
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

Serial.println("Waiting for Firebase...");

while (!Firebase.ready()) {
    delay(100);
}

Serial.println("Firebase Ready");

// Restore previous progress

loadSavedData();
}

// =====
// LOOP
// =====

void loop() {

    // Check if parent changed goal

    readGoal();

    // Read button states

    bool b1 = digitalRead(BTN_1);
    bool b2 = digitalRead(BTN_2);

```

```

bool b5  = digitalRead(BTN_5);
bool b10 = digitalRead(BTN_10);

// Detect HIGH -> LOW transition
// (button press)

if (lastBtn1 == HIGH && b1 == LOW) {
    addAmount(1);
}

if (lastBtn2 == HIGH && b2 == LOW) {
    addAmount(2);
}

if (lastBtn5 == HIGH && b5 == LOW) {
    addAmount(5);
}

if (lastBtn10 == HIGH && b10 == LOW) {
    addAmount(10);
}

// Store current states

lastBtn1  = b1;
lastBtn2  = b2;
lastBtn5  = b5;
lastBtn10 = b10;

delay(50);
}

// =====
// LOAD SAVED DATA
//
// Called once during startup

```

```
// =====

void loadSavedData() {

    if (!Firebase.ready()) return;

    // Restore total

    if (Firebase.RTDB.getInt(&fbdo, "/piggy/total")) {

        totalAmount = fbdo.intData();

        Serial.print("Loaded Total: ");
        Serial.println(totalAmount);
    }

    // Restore goal
    // Kodular stores goal as STRING

    if (Firebase.RTDB.get(&fbdo, "/piggy/goal")) {

        if (fbdo.dataType() == "string")
            goal = fbdo.stringData().toInt();

        else if (fbdo.dataType() == "int")
            goal = fbdo.intData();

        else if (fbdo.dataType() == "float")
            goal = (int)fbdo.floatData();

        else if (fbdo.dataType() == "double")
            goal = (int)fbdo.doubleData();

        lastGoal = goal;

        Serial.print("Loaded Goal: ");
    }
}
```

```

        Serial.println(goal);
    }
}

// =====
// READ GOAL
//
// Runs every 2 seconds
//
// If parent changed goal:
// reset challenge
// =====

void readGoal() {

    if (!Firebase.ready()) return;

    if (millis() - lastFirebaseRead < firebaseInterval)
        return;

    lastFirebaseRead = millis();

    if (Firebase.RTDB.get(&fbdo, "/piggy/goal")) {

        int newGoal = 0;

        if (fbdo.dataType() == "string")
            newGoal = fbdo.stringData().toInt();

        else if (fbdo.dataType() == "int")
            newGoal = fbdo.intData();

        else if (fbdo.dataType() == "float")
            newGoal = (int)fbdo.floatData();

        else if (fbdo.dataType() == "double")

```

```

        newGoal = (int)fbdo.doubleData();

// Parent changed goal

if (newGoal != lastGoal) {

    goal = newGoal;
    lastGoal = newGoal;

    Serial.println();
    Serial.println("==== NEW GOAL DETECTED ====");
    Serial.print("Goal = ");
    Serial.println(goal);

// Start new challenge

    totalAmount = 0;

    Firebase.RTDB.setInt(&fbdo, "/piggy/total", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", 0);

    Serial.println("Savings Reset");
}
}
}

// =====
// ADD MONEY
// =====

void addAmount(int value) {

    totalAmount += value;

    int percentage = 0;

```

```

    if (goal > 0) {
        percentage = (totalAmount * 100) / goal;
    }

    Serial.print("Added ₹");
    Serial.print(value);

    Serial.print(" | Total ₹");
    Serial.print(totalAmount);

    Serial.print(" | ");
    Serial.print(percentage);
    Serial.println("%");

    sendToFirebase(totalAmount, percentage, value);
}

// =====
// UPDATE FIREBASE
//
// ESP32 NEVER WRITES GOAL
// App owns the goal
// =====

void sendToFirebase(int total, int percentage, int lastCoin)
{

    if (!Firebase.ready()) return;

    Firebase.RTDB.setInt(&fbdo, "/piggy/total", total);
    Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", percentage);
    Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", lastCoin);
}

```

KODULAR+FIREBASE+BUTTON INPUTS+JQ6500+SOLENOID(CHUTE)

```
// =====  
// PENNY PAL  
// XIAO ESP32-C6  
//  
// B1 = D1 = Rs5  
// B2 = D2 = Rs1  
// B3 = D5 = Rs10  
// B4 = D3 = Rs2  
// D8 = IR sensor  
// D10 = Solenoid  
// D7 (GPIO17) = MP3 RX ← JQ6500 TX  
// D6 (GPIO16) = MP3 TX → JQ6500 RX  
//  
// MP3 track map:  
// Track 1 = coin detected sound (IR beam break)  
// Track 2 = coin insert sound (button press)  
// Track 3 = 25% milestone  
// Track 4 = goal achieved  
//  
// Parent App:  
// writes /piggy/goal  
//  
// ESP32:  
// reads /piggy/goal  
// writes /piggy/total  
// writes /piggy/percentage  
// writes /piggy/lastCoin
```

```

//
// New Goal:
// resets savings challenge
//
// Reboot:
// restores progress from Firebase
// =====
// ----- WIFI -----
#include <WiFi.h>

// ----- FIREBASE -----
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"

// ----- MP3 -----
#include <JQ6500_Serial.h>

// ----- WIFI CREDENTIALS -----
const char* ssid    = "MiCho";
const char* password = "Hello1234";

// ----- FIREBASE CONFIG -----
#define API_KEY      "AlzaSyDpefz07KOUYyj0mZ1YtvJkm5g3z5HU0-I"
#define DATABASE_URL "https://smart-piggy-effc6-default-rtdb.asia-southeast1.firebaseio.com/"

FirebaseData  fbdo;
FirebaseAuth  auth;
FirebaseConfig config;

// =====
// BUTTON PINS
//
// B1 = D1 = Rs5
// B2 = D2 = Rs1
// B3 = D5 = Rs10

```

```

// B4 = D3 = Rs2
// =====

#define BTN_1  D1  // Rs5
#define BTN_2  D2  // Rs1
#define BTN_3  D5  // Rs10
#define BTN_4  D3  // Rs2

// =====
// IR SENSOR PIN
// =====

#define IR_SENSOR  D8
#define COIN_DETECTED HIGH

// =====
// SOLENOID PIN
// =====

#define SOLENOID  D10

// =====
// MP3 PINS (JQ6500)
// =====

#define MP3_RX_PIN 17  // D7 → JQ6500 TX
#define MP3_TX_PIN 16  // D6 → JQ6500 RX

HardwareSerial mp3Serial(1);
JQ6500_Serial mp3(mp3Serial);

// =====
// SAVINGS DATA
// =====

int totalAmount = 0;
int goal        = 0;
int lastGoal    = -1;

// =====
// MILESTONE FLAGS
// =====

```

```

bool quarterPlayed = false;
bool goalPlayed    = false;

// =====
// IR SENSOR STATE
// =====

bool coinPresent      = false;
int  lastIRState      = LOW;
unsigned long lastTriggerTime = 0;
const int IR_LOCKOUT   = 200;

// =====
// FIREBASE POLL TIMER
// =====

unsigned long lastFirebaseRead    = 0;
const unsigned long firebaseInterval = 2000;

// =====
// BUTTON STATE MEMORY
//
// INPUT_PULLUP:
//  HIGH = not pressed
//  LOW  = pressed
// =====

bool lastBtn1 = HIGH;
bool lastBtn2 = HIGH;
bool lastBtn3 = HIGH;
bool lastBtn4 = HIGH;

// =====
// SETUP
// =====

void setup() {
  Serial.begin(115200);

  Serial.println();
  Serial.println("=====");

```

```

Serial.println("PENNY PAL STARTING");
Serial.println("=====");

// Button pins

pinMode(BTN_1, INPUT_PULLUP);
pinMode(BTN_2, INPUT_PULLUP);
pinMode(BTN_3, INPUT_PULLUP);
pinMode(BTN_4, INPUT_PULLUP);

// IR sensor

pinMode(IR_SENSOR, INPUT);
lastIRState = digitalRead(IR_SENSOR);

// Solenoid

pinMode(SOLENOID, OUTPUT);
digitalWrite(SOLENOID, LOW);

// ----- WIFI -----

Serial.print("Connecting WiFi");
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}

Serial.println();
Serial.println("WiFi Connected");
Serial.print("IP: ");
Serial.println(WiFi.localIP());

// ----- FIREBASE -----

config.api_key          = API_KEY;
config.database_url      = DATABASE_URL;
config.token_status_callback = tokenStatusCallback;

Firebase.signUp(&config, &auth, "", "");
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

```

```

Serial.println("Waiting for Firebase...");
while (!Firebase.ready()) {
  delay(100);
}

Serial.println("Firebase Ready");

// Restore previous progress
loadSavedData();

// ----- MP3 -----

mp3Serial.begin(9600, SERIAL_8N1, MP3_RX_PIN, MP3_TX_PIN);
delay(3000);
mp3.reset();
delay(1000);
mp3.setSource(MP3_SRC_BUILTIN);
delay(500);
mp3.setVolume(30);

Serial.println("MP3 Ready");
Serial.println("=====");
}

// =====
// LOOP
// =====

void loop() {
  // Check if parent changed goal
  readGoal();

  // ---- IR SENSOR ----

  int currentIR = digitalRead(IR_SENSOR);

  if (lastIRState == LOW && currentIR == COIN_DETECTED) {

```

```

    if (millis() - lastTriggerTime > IR_LOCKOUT) {

      coinPresent      = true;
      lastTriggerTime = millis();
    }
  }
}

```

```
    Serial.println("Coin Detected – waiting for denomination button");

    mp3.playFileByIndexNumber(1);
}
```

```
}
```

```
if (currentIR != COIN_DETECTED) {
    coinPresent = false;
}
```

```
lastIRState = currentIR;
```

```
// ---- BUTTONS ----
```

```
// Only act if a coin is present in the slot
```

```
bool b1 = digitalRead(BTN_1);
```

```
bool b2 = digitalRead(BTN_2);
```

```
bool b3 = digitalRead(BTN_3);
```

```
bool b4 = digitalRead(BTN_4);
```

```
if (coinPresent) {
```

```
    if (lastBtn1 == HIGH && b1 == LOW) {
        addAmount(5);        // B1 = Rs5
        coinPresent = false;
    }
```

```
    if (lastBtn2 == HIGH && b2 == LOW) {
        addAmount(1);        // B2 = Rs1
        coinPresent = false;
    }
```

```
    if (lastBtn3 == HIGH && b3 == LOW) {
        addAmount(10);       // B3 = Rs10
        coinPresent = false;
    }
```

```
    if (lastBtn4 == HIGH && b4 == LOW) {
        addAmount(2);        // B4 = Rs2
        coinPresent = false;
    }
```

```
} else {
```

```

    if ((lastBtn1 == HIGH && b1 == LOW) ||
        (lastBtn2 == HIGH && b2 == LOW) ||
        (lastBtn3 == HIGH && b3 == LOW) ||
        (lastBtn4 == HIGH && b4 == LOW)) {

        Serial.println("Button ignored – no coin detected");
    }

}

// Store button states

lastBtn1 = b1;
lastBtn2 = b2;
lastBtn3 = b3;
lastBtn4 = b4;

delay(50);
}

// =====
// LOAD SAVED DATA
//
// Called once during startup
// =====

void loadSavedData() {
    if (!Firebase.ready()) return;

    if (Firebase.RTDB.getInt(&fbdo, "/piggy/total")) {
        totalAmount = fbdo.intData();
        Serial.print("Loaded Total: ");
        Serial.println(totalAmount);
    }

    if (Firebase.RTDB.get(&fbdo, "/piggy/goal")) {

```

```

        if (fbdo.dataType() == "string")
            goal = fbdo.stringData().toInt();
        else if (fbdo.dataType() == "int")
            goal = fbdo.intData();
        else if (fbdo.dataType() == "float")

```

```

        goal = (int)fbdo.floatData();
    else if (fbdo.dataType() == "double")
        goal = (int)fbdo.doubleData();

    lastGoal = goal;

    Serial.print("Loaded Goal: ");
    Serial.println(goal);

}

if (goal > 0) {
    if (totalAmount >= (goal / 4)) quarterPlayed = true;
    if (totalAmount >= goal)    goalPlayed    = true;
}
}

// =====
// READ GOAL
//
// Runs every 2 seconds
// Resets challenge if parent changed goal
// =====

void readGoal() {
    if (!Firebase.ready()) return;

    if (millis() - lastFirebaseRead < firebaseInterval)
        return;

    lastFirebaseRead = millis();

    if (Firebase.RTDB.get(&fbdo, "/piggy/goal")) {

```

```

        int newGoal = 0;

        if (fbdo.dataType() == "string")
            newGoal = fbdo.stringData().toInt();
        else if (fbdo.dataType() == "int")
            newGoal = fbdo.intData();
        else if (fbdo.dataType() == "float")
            newGoal = (int)fbdo.floatData();
        else if (fbdo.dataType() == "double")
            newGoal = (int)fbdo.doubleData();
    }
}

```

```

if (newGoal != lastGoal) {

    goal      = newGoal;
    lastGoal = newGoal;

    Serial.println();
    Serial.println("==== NEW GOAL DETECTED =====");
    Serial.print("Goal = ");
    Serial.println(goal);

    totalAmount = 0;
    quarterPlayed = false;
    goalPlayed = false;

    Firebase.RTDB.setInt(&fbdo, "/piggy/total", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", 0);

    Serial.println("Savings Reset");
}

```

```

}
}

// =====
// ADD MONEY
//
// Only called after IR confirms coin is present
// =====

void addAmount(int value) {
    mp3.playFileByIndexNumber(2);
    delay(500);
    pulseSolenoid();
    totalAmount += value;
    int percentage = 0;
    if (goal > 0) {
        percentage = (totalAmount * 100) / goal;
    }
}

```

```

Serial.print("Added Rs");
Serial.print(value);
Serial.print(" | Total Rs");
Serial.print(totalAmount);
Serial.print(" | ");
Serial.print(percentage);
Serial.println("%");

sendToFirebase(totalAmount, percentage, value);

checkMilestones();
}

// =====
// PULSE SOLENOID
// =====

void pulseSolenoid() {
digitalWrite(SOLENOID, HIGH);
delay(100);
digitalWrite(SOLENOID, LOW);
}

// =====
// CHECK MILESTONES
// =====

void checkMilestones() {
if (goal <= 0) return;

if (!quarterPlayed && totalAmount >= (goal / 4)) {
quarterPlayed = true;
Serial.println("25% Goal Reached!");
mp3.playFileByIndexNumber(3);
}

if (!goalPlayed && totalAmount >= goal) {
goalPlayed = true;
Serial.println("GOAL ACHIEVED!");
mp3.playFileByIndexNumber(4);
}

```

```

}
}

// =====
// UPDATE FIREBASE
//
// ESP32 NEVER WRITES GOAL — app owns it
// =====

void sendToFirebase(int total, int percentage, int lastCoin) {
  if (!Firebase.ready()) return;

  Firebase.RTDB.setInt(&fbdo, "/piggy/total", total);
  Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", percentage);
  Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", lastCoin);
}

```

neopixel test: numbers 0 to 5 on serial monitor lights up the following:

0 lights up 10 rupee, 1 lights up 2 rupee, 2 lights up 1 rupee and 3 lights up 5 rupee. and 4 and 5 are for chamber lights.

```

#include <Adafruit_NeoPixel.h>

#define NEO_PIN D4
#define NEO_COUNT 6

Adafruit_NeoPixel strip(NEO_COUNT, NEO_PIN, NEO_GRB + NEO_KHZ
800);

void setup() {
  Serial.begin(115200);

  strip.begin();
}

```

```

strip.setBrightness(100);
strip.clear();
strip.show();

Serial.println("Type 0-5 to light a pixel, or 'all' to light all, or 'off' to clear");
}

void loop() {

    if (Serial.available()) {

        String input = Serial.readStringUntil('\n');
        input.trim();

        if (input == "all") {

            for (int i = 0; i < NEO_COUNT; i++) {
                strip.setPixelColor(i, strip.Color(255, 0, 0));
            }
            strip.show();
            Serial.println("All ON");

        } else if (input == "off") {

            strip.clear();
            strip.show();
            Serial.println("All OFF");

        } else {

            int index = input.toInt();

            if (index >= 0 && index < NEO_COUNT) {
                strip.clear();
                strip.setPixelColor(index, strip.Color(255, 0, 0));
            }
        }
    }
}

```

```

        strip.show();
        Serial.print("Pixel ");
        Serial.print(index);
        Serial.println(" ON");
    } else {
        Serial.println("Invalid - enter 0 to 5, 'all', or 'of
f'");
    }
}
}
}
}
}

```

Neopixel integrated

```

// =====
// PENNY PAL
// XIAO ESP32-C6
//
// BUTTONS:
//   B1 = D1  = Rs5
//   B2 = D2  = Rs1
//   B3 = D5  = Rs10
//   B4 = D3  = Rs2
//
// NEOPIXEL (D4, 6 total):
//   Index 0 = Rs10 (Gold)
//   Index 1 = Rs2  (Green)
//   Index 2 = Rs1  (White)
//   Index 3 = Rs5  (Blue)
//   Index 4 = Chamber light
//   Index 5 = Chamber light

```

```

//
// OTHER:
//   D8   = IR sensor
//   D10  = Solenoid
//   D7   (GPIO17) = MP3 RX <- JQ6500 TX
//   D6   (GPIO16) = MP3 TX -> JQ6500 RX
//
// MP3 TRACKS:
//   1 = coin detected (IR)
//   2 = coin accepted (button)
//   3 = 25% milestone
//   4 = goal achieved
//
// FIREBASE:
//   App writes -> /piggy/goal
//   ESP32 reads <- /piggy/goal
//   ESP32 writes-> /piggy/total
//                   /piggy/percentage
//                   /piggy/lastCoin
// =====

#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
#include <JQ6500_Serial.h>
#include <Adafruit_NeoPixel.h>

// =====
// WIFI + FIREBASE
// =====

const char* WIFI_SSID   = "MiCho";
const char* WIFI_PASS   = "Hello1234";

#define API_KEY          "AIzaSyDpefz07K0uYyj0mZ1YtvJkm5g3z5HU0-"

```

```

I"
#define DATABASE_URL "https://smart-piggy-effc6-default-rtdb.
asia-southeast1.firebaseio.com/"

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// =====
// PINS
// =====

#define BTN_1      D1      // Rs5
#define BTN_2      D2      // Rs1
#define BTN_3      D5      // Rs10
#define BTN_4      D3      // Rs2
#define IR_SENSOR  D8
#define SOLENOID   D10
#define NEO_PIN    D4
#define MP3_RX_PIN 17
#define MP3_TX_PIN 16

// =====
// NEOPIXEL
// =====

#define NEO_COUNT  6
#define NEO_BUTTONS 4 // indices 0-3 are buttons, 4-5 are
chamber

Adafruit_NeoPixel strip(NEO_COUNT, NEO_PIN, NEO_GRB + NEO_KHZ
800);

// Button -> LED index mapping
// { btn index, led index, R, G, B }
const uint8_t LED_MAP[4][5] = {

```

```

    { 0, 3, 0, 0, 255 }, // B1 Rs5 -> LED 3 Blue
    { 1, 2, 255, 255, 255 }, // B2 Rs1 -> LED 2 White
    { 2, 0, 255, 180, 0 }, // B3 Rs10 -> LED 0 Gold
    { 3, 1, 0, 255, 0 }, // B4 Rs2 -> LED 1 Green
};

// =====
// MP3
// =====

HardwareSerial mp3Serial(1);
JQ6500_Serial mp3(mp3Serial);

// =====
// SAVINGS
// =====

int totalAmount = 0;
int goal         = 0;
int lastGoal     = -1;

// =====
// FLAGS
// =====

bool quarterPlayed = false;
bool goalPlayed    = false;
bool coinPresent   = false;

// =====
// BUTTON DENOMINATIONS
// =====

const int DENOM[4] = { 5, 1, 10, 2 }; // B1, B2, B3, B4

// =====

```

```

// BUTTON STATE
// =====

bool lastBtn[4] = { HIGH, HIGH, HIGH, HIGH };
const uint8_t BTN_PINS[4] = { BTN_1, BTN_2, BTN_3, BTN_4 };

// =====
// IR
// =====

int lastIRState          = LOW;
unsigned long lastTriggerTime = 0;
const int IR_LOCKOUT      = 200;

// =====
// PULSE ANIMATION (non-blocking)
// =====

bool      pulsing          = false;
int       pulseStep        = 0;
int       pulseCycleCount  = 0;
unsigned long lastPulseTime = 0;
const int PULSE_STEPS      = 8;
const int PULSE_DELAY      = 60;
const int PULSE_CYCLES     = 3;

// =====
// FIREBASE TIMER
// =====

unsigned long lastFirebaseRead = 0;
const unsigned long FIREBASE_INTERVAL = 2000;

// =====
// HELPERS – NEOPIXEL
// =====

```

```

void clearButtonLEDs() {
    for (int i = 0; i < NEO_BUTTONS; i++) {
        strip.setPixelColor(i, 0);
    }
    strip.show();
}

void flashLED(int ledIndex, uint8_t r, uint8_t g, uint8_t b)
{
    clearButtonLEDs();
    strip.setPixelColor(ledIndex, strip.Color(r, g, b));
    strip.show();
    delay(400);
    clearButtonLEDs();
}

void startPulse() {
    pulsing      = true;
    pulseStep    = 0;
    pulseCycleCount = 0;
    lastPulseTime = millis();
}

void stopPulse() {
    pulsing = false;
    clearButtonLEDs();
}

void updatePulse() {

    if (!pulsing) return;
    if (millis() - lastPulseTime < PULSE_DELAY) return;

    lastPulseTime = millis();
}

```

```

int totalSteps = PULSE_STEPS * 2;
int brightness = (pulseStep < PULSE_STEPS)
    ? map(pulseStep, 0, PULSE_STEPS, 0, 255)
    : map(pulseStep, PULSE_STEPS, totalSteps, 255, 0);

uint32_t c = strip.Color(brightness, brightness, brightnes
s);

for (int i = 0; i < NEO_BUTTONS; i++) {
    strip.setPixelColor(i, c);
}
strip.show();

if (++pulseStep >= totalSteps) {
    pulseStep = 0;
    if (++pulseCycleCount >= PULSE_CYCLES) {
        stopPulse();
    }
}
}

// =====
// HELPERS – FIREBASE
// =====

int parseFirebaseInt(FirebaseData& fd) {
    if (fd.dataType() == "string")        return fd.stringData
().toInt();
    else if (fd.dataType() == "int")      return fd.intData();
    else if (fd.dataType() == "float")    return (int)fd.floatDa
ta();
    else if (fd.dataType() == "double")  return (int)fd.doublEd
ata();
    return 0;
}

```

```

void sendToFirebase(int total, int percentage, int lastCoin)
{
    if (!Firebase.ready()) return;
    Firebase.RTDB.setInt(&fbdo, "/piggy/total", total);
    Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", percentage);
    Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", lastCoin);
}

// =====
// LOAD SAVED DATA
// =====

void loadSavedData() {

    if (!Firebase.ready()) return;

    if (Firebase.RTDB.getInt(&fbdo, "/piggy/total")) {
        totalAmount = fbdo.intData();
        Serial.print("Loaded Total: Rs");
        Serial.println(totalAmount);
    }

    if (Firebase.RTDB.get(&fbdo, "/piggy/goal")) {
        goal = parseFirebaseInt(fbdo);
        lastGoal = goal;
        Serial.print("Loaded Goal: Rs");
        Serial.println(goal);
    }

    // Restore milestone flags – prevents replaying on reboot

    if (goal > 0) {
        if (totalAmount >= (goal / 4)) quarterPlayed = true;
        if (totalAmount >= goal) goalPlayed = true;
    }
}

```

```

}

// =====
// READ GOAL (every 2 seconds)
// =====

void readGoal() {

    if (!Firebase.ready()) return;
    if (millis() - lastFirebaseRead < FIREBASE_INTERVAL) return;

    lastFirebaseRead = millis();

    if (!Firebase.RTDB.get(&fbdo, "/piggy/goal")) return;

    int newGoal = parseFirebaseInt(fbdo);

    if (newGoal == lastGoal) return;

    goal = newGoal;
    lastGoal = newGoal;

    totalAmount = 0;
    quarterPlayed = false;
    goalPlayed = false;

    Firebase.RTDB.setInt(&fbdo, "/piggy/total", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/percentage", 0);
    Firebase.RTDB.setInt(&fbdo, "/piggy/lastCoin", 0);

    Serial.print("New Goal: Rs");
    Serial.println(goal);
    Serial.println("Savings Reset");
}

```

```

// =====
// CHECK MILESTONES
// =====

void checkMilestones() {

    if (goal <= 0) return;

    if (!quarterPlayed && totalAmount >= (goal / 4)) {
        quarterPlayed = true;
        Serial.println("25% Reached!");
        mp3.playFileByIndexNumber(3);
    }

    if (!goalPlayed && totalAmount >= goal) {
        goalPlayed = true;
        Serial.println("Goal Achieved!");
        mp3.playFileByIndexNumber(4);
    }
}

// =====
// ADD MONEY
// =====

void addAmount(int btnIndex) {

    int value = DENOM[btnIndex];

    mp3.playFileByIndexNumber(2);

    delay(500);

    digitalWrite(SOLENOID, HIGH);
    delay(100);
    digitalWrite(SOLENOID, LOW);
}

```

```

    totalAmount += value;

    int percentage = (goal > 0) ? (totalAmount * 100) / goal :
0;

    Serial.print("Added Rs");
    Serial.print(value);
    Serial.print(" | Total Rs");
    Serial.print(totalAmount);
    Serial.print(" | ");
    Serial.print(percentage);
    Serial.println("%");

    sendToFirebase(totalAmount, percentage, value);
    checkMilestones();
}

// =====
// SETUP
// =====

void setup() {

    Serial.begin(115200);
    Serial.println("\n=====");
    Serial.println("PENNY PAL STARTING");
    Serial.println("=====");

    // Buttons

    for (int i = 0; i < 4; i++) {
        pinMode(BTN_PINS[i], INPUT_PULLUP);
    }

    // IR + Solenoid

```

```

pinMode(IR_SENSOR, INPUT);
lastIRState = digitalRead(IR_SENSOR);

pinMode(SOLENOID, OUTPUT);
digitalWrite(SOLENOID, LOW);

// NeoPixel

strip.begin();
strip.setBrightness(180);
strip.clear();
strip.show();

// WiFi

Serial.print("Connecting WiFi");
WiFi.begin(WIFI_SSID, WIFI_PASS);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}

Serial.println("\nWiFi Connected - IP: " + WiFi.localIP().toString());

// Firebase

config.api_key          = API_KEY;
config.database_url      = DATABASE_URL;
config.token_status_callback = tokenStatusCallback;

Firebase.signUp(&config, &auth, "", "");
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);

```

```

Serial.println("Waiting for Firebase...");
while (!Firebase.ready()) { delay(100); }
Serial.println("Firebase Ready");

loadSavedData();

// MP3

mp3Serial.begin(9600, SERIAL_8N1, MP3_RX_PIN, MP3_TX_PIN);
delay(3000);
mp3.reset();
delay(1000);
mp3.setSource(MP3_SRC_BUILTIN);
delay(500);
mp3.setVolume(30);

Serial.println("MP3 Ready");
Serial.println("=====");
}

// =====
// LOOP
// =====

void loop() {

    readGoal();

    // ---- IR ----

    int currentIR = digitalRead(IR_SENSOR);

    if (lastIRState == LOW && currentIR == HIGH) {
        if (millis() - lastTriggerTime > IR_LOCKOUT) {
            coinPresent = true;

```

```

        lastTriggerTime = millis();
        Serial.println("Coin detected – select denomination");
        mp3.playFileByIndexNumber(1);
        startPulse();
    }
}

if (currentIR == LOW) {
    if (coinPresent) {
        coinPresent = false;
        stopPulse();
    }
}

lastIRState = currentIR;

updatePulse();

// ---- BUTTONS ----

for (int i = 0; i < 4; i++) {

    bool current = digitalRead(BTN_PINS[i]);

    if (lastBtn[i] == HIGH && current == LOW) {

        if (coinPresent) {
            stopPulse();
            flashLED(LED_MAP[i][1], LED_MAP[i][2], LED_MAP[i][3],
LED_MAP[i][4]);
            addAmount(i);
            coinPresent = false;
        } else {
            Serial.println("Button ignored – no coin");
        }
    }
}

```

```
    lastBtn[i] = current;  
}  
  
delay(50);  
}
```