**SMART RFID BASED PAYMENT SYSTEM:**

**#include <SPI.h>**

**#include <MFRC522.h>**

**#include <Arduino.h>**

**#include <ESP8266WiFi.h>**

**#include <Firebase\_ESP\_Client.h>**

**#include <ArduinoJson.h> // For parsing JSON data**

**#include <addons/TokenHelper.h>**

**#include <addons/RTDBHelper.h>**

**// RFID Setup**

**#define SS\_PIN 2 // SDA**

**#define RST\_PIN 0 // RST**

**MFRC522 rfid(SS\_PIN, RST\_PIN);**

**// WiFi and Firebase Setup**

**#define WIFI\_SS "Galaxy A10s5584"**

**#define WIFI\_PASSWORD "isaitharani"**

**#define API\_KEY "AIzaSyCi-2z5tjB8LLBbMCZeuewGi6tSQtF9iPg"**

**#define DATABASE\_URL "https://smart-way-payment-default-rtdb.firebaseio.com"**

**#define USER\_EMAIL "thirisha1227@gmail.com"**

**#define USER\_PASSWORD "Thirisha@2711"**

**FirebaseData fbdo;**

**FirebaseAuth auth;**

**FirebaseConfig config;**

**// Updated UIDs for lookup**

**const char\* uids[] = {"af14e91c", "3351e534", "8f13ea1f"}; // Add your UIDs for lookup**

**void setup()**

**{**

**Serial.begin(115200);**

**// Initialize RFID**

**SPI.begin();**

**rfid.PCD\_Init();**

**Serial.println("Scan an RFID tag...");**

**// Connect to Wi-Fi**

**WiFi.begin(WIFI\_SS, WIFI\_PASSWORD);**

**Serial.print("Connecting to Wi-Fi");**

**while (WiFi.status() != WL\_CONNECTED)**

**{**

**Serial.print(".");**

**delay(300);**

**}**

**Serial.println();**

**Serial.print("Connected with IP: ");**

**Serial.println(WiFi.localIP());**

**Serial.println();**

**// Set up Firebase configuration**

**config.api\_key = API\_KEY;**

**auth.user.email = USER\_EMAIL;**

**auth.user.password = USER\_PASSWORD;**

**config.database\_url = DATABASE\_URL;**

**config.token\_status\_callback = tokenStatusCallback; // For debugging purposes**

**Firebase.begin(&config, &auth);**

**Firebase.reconnectWiFi(true);**

**}**

**void loop()**

**{**

**// Check for RFID card presence**

**if (!rfid.PICC\_IsNewCardPresent())**

**{**

**return;**

**}**

**// Read RFID card**

**if (!rfid.PICC\_ReadCardSerial())**

**{**

**return;**

**}**

**// Get UID as a string**

**String scannedUID = "";**

**for (byte i = 0; i < rfid.uid.size; i++) {**

**scannedUID += (rfid.uid.uidByte[i] < 0x10 ? "0" : "") + String(rfid.uid.uidByte[i], HEX);**

**}**

**Serial.println("Scanned UID: " + scannedUID);**

**// Match the scanned UID with predefined UIDs**

**for (int i = 0; i < sizeof(uids) / sizeof(uids[0]); i++)**

**{**

**if (scannedUID.equalsIgnoreCase(uids[i]))**

**{**

**Serial.println("User Found: " + String(uids[i]));**

**fetchData(uids[i]); // Fetch data for the matched UID**

**break;**

**}**

**}**

**// Halt PICC**

**rfid.PICC\_HaltA();**

**}**

**void fetchData(const char\* uid) {**

**// Set the paths for amount and name**

**String pathAmount = "UID/" + String(uid) + "/amount";**

**String pathName = "UID/" + String(uid) + "/name";**

**// Fetch and print the amount**

**printAmount(pathAmount.c\_str(), uid);**

**// Fetch and print the name**

**printName(pathName.c\_str(), uid);**

**}**

**void printAmount(const char\* path, const char\* uid)**

**{**

**// Retrieve and print the amount associated with the UID**

**if (Firebase.RTDB.getInt(&fbdo, path))**

**{**

**int amount = fbdo.intData(); // Assign amount from Firebase**

**Serial.println("Amount for " + String(uid) + ": " + String(amount)); // Print amount on a separate line**

**}**

**else**

**{**

**Serial.println("Error getting amount for " + String(uid) + ": " + fbdo.errorReason());**

**}**

**}**

**void printName(const char\* path, const char\* uid)**

**{**

**// Retrieve and print the name associated with the UID**

**if (Firebase.RTDB.get(&fbdo, path))**

**{**

**Serial.print("Raw data for name: " + String(fbdo.stringData())); // Debug output**

**if (fbdo.dataType() == "string")**

**{**

**const char\* name = fbdo.stringData().c\_str(); // Use stringData() for string retrieval**

**Serial.println("Name for " + String(uid) + ": " + String(name)); // Print name on a separate line**

**}**

**else**

**{**

**Serial.println("Name data is not a string for " + String(uid));**

**}**

**}**

**else**

**{**

**Serial.println("Error getting name for " + String(uid) + ": " + fbdo.errorReason());**

**}**

**}**

**OUTPUT:**

**20:23:19.285 -> ;l$��<�d�|�l�c|����{�c�c��'o�$go���c8��ds$s$p�n��l��cg�|���c��ng�lć$`�go$`n;Ǜ�n#d�lx�o�s�ܜ���bn�<�#��o'�l`�ogl`gs���gc��`;��gc��`��|�$`��'�l����o�r��'|�$�d`c��|{�l�o��g�d`��s�l�l��Scan an RFID tag...**

**20:23:19.286 -> Connecting to Wi-Fi...........**

**20:23:23.001 -> Connected with IP: 192.168.43.120**

**20:23:23.118 ->**

**20:23:23.118 -> Token info: type = id token (GITKit token), status = on request**

**20:23:25.327 -> Token info: type = id token (GITKit token), status = ready**

**20:23:25.840 -> Scanned UID: 8f13ea1f**

**20:23:25.840 -> User Found: 8f13ea1f**

**20:23:28.337 -> Amount for 8f13ea1f: 6000**

**20:23:28.863 -> Raw data for name: SowndaryaName for 8f13ea1f:**

**20:23:31.297 -> Scanned UID: 3351e534**

**20:23:31.297 -> User Found: 3351e534**

**20:23:31.792 -> Amount for 3351e534: 5000**

**20:23:32.313 -> Raw data for name: TharaniName for 3351e534:**

**20:23:36.837 -> Scanned UID: af14e91c**

**20:23:36.837 -> User Found: af14e91c**

**20:23:37.370 -> Amount for af14e91c: 7000**

**20:23:37.927 -> Raw data for name: ThrishaName for af14e91c:**

**CLOUD DATA FETCHING TAPED RFID PIN:**

**#include <SPI.h>**

**#include <MFRC522.h>**

**#include <Arduino.h>**

**#include <ESP8266WiFi.h>**

**#include <Firebase\_ESP\_Client.h>**

**#include <ArduinoJson.h> // For parsing JSON data**

**#include <addons/TokenHelper.h>**

**#include <addons/RTDBHelper.h>**

**// RFID Setup**

**#define SS\_PIN 2 // SDA**

**#define RST\_PIN 0 // RST**

**MFRC522 rfid(SS\_PIN, RST\_PIN);**

**// WiFi and Firebase Setup**

**#define WIFI\_SS "Galaxy A10s5584"**

**#define WIFI\_PASSWORD "isaitharani"**

**#define API\_KEY "AIzaSyCi-2z5tjB8LLBbMCZeuewGi6tSQtF9iPg"**

**#define DATABASE\_URL "https://smart-way-payment-default-rtdb.firebaseio.com"**

**#define USER\_EMAIL "thirisha1227@gmail.com"**

**#define USER\_PASSWORD "Thirisha@2711"**

**FirebaseData fbdo;**

**FirebaseAuth auth;**

**FirebaseConfig config;**

**// Updated UIDs for lookup**

**const char\* uids[] = {"af14e91c", "3351e534", "8f13ea1f"}; // Add your UIDs for lookup**

**void setup()**

**{**

**Serial.begin(115200);**

**// Initialize RFID**

**SPI.begin();**

**rfid.PCD\_Init();**

**Serial.println("Scan an RFID tag...");**

**// Connect to Wi-Fi**

**WiFi.begin(WIFI\_SS, WIFI\_PASSWORD);**

**Serial.print("Connecting to Wi-Fi");**

**while (WiFi.status() != WL\_CONNECTED)**

**{**

**Serial.print(".");**

**delay(300);**

**}**

**Serial.println();**

**Serial.print("Connected with IP: ");**

**Serial.println(WiFi.localIP());**

**Serial.println();**

**// Set up Firebase configuration**

**config.api\_key = API\_KEY;**

**auth.user.email = USER\_EMAIL;**

**auth.user.password = USER\_PASSWORD;**

**config.database\_url = DATABASE\_URL;**

**config.token\_status\_callback = tokenStatusCallback; // For debugging purposes**

**Firebase.begin(&config, &auth);**

**Firebase.reconnectWiFi(true);**

**}**

**void loop()**

**{**

**// Check for RFID card presence**

**if (!rfid.PICC\_IsNewCardPresent())**

**{**

**return;**

**}**

**// Read RFID card**

**if (!rfid.PICC\_ReadCardSerial())**

**{**

**return;**

**}**

**// Get UID as a string**

**String scannedUID = "";**

**for (byte i = 0; i < rfid.uid.size; i++) {**

**scannedUID += (rfid.uid.uidByte[i] < 0x10 ? "0" : "") + String(rfid.uid.uidByte[i], HEX);**

**}**

**Serial.println("Scanned UID: " + scannedUID);**

**// Match the scanned UID with predefined UIDs**

**for (int i = 0; i < sizeof(uids) / sizeof(uids[0]); i++)**

**{**

**if (scannedUID.equalsIgnoreCase(uids[i]))**

**{**

**Serial.println("User Found: " + String(uids[i]));**

**fetchData(uids[i]); // Fetch data for the matched UID**

**break;**

**}**

**}**

**// Halt PICC**

**rfid.PICC\_HaltA();**

**}**

**void fetchData(const char\* uid) {**

**// Set the paths for amount and name**

**String pathAmount = "UID/" + String(uid) + "/amount";**

**String pathName = "UID/" + String(uid) + "/name";**

**String pathpin = "UID/" + String(uid) + "/pin";**

**// Fetch and print the amount**

**printAmount(pathAmount.c\_str(), uid);**

**// Fetch and print the name**

**printName(pathName.c\_str(), uid);**

**printpin(pathpin.c\_str(), uid);**

**}**

**void printAmount(const char\* path, const char\* uid)**

**{**

**// Retrieve and print the amount associated with the UID**

**if (Firebase.RTDB.getInt(&fbdo, path))**

**{**

**int amount = fbdo.intData(); // Assign amount from Firebase**

**Serial.println("Amount for " + String(uid) + ": " + String(amount)); // Print amount on a separate line**

**}**

**else**

**{**

**Serial.println("Error getting amount for " + String(uid) + ": " + fbdo.errorReason());**

**}**

**}**

**void printName(const char\* path, const char\* uid)**

**{**

**// Retrieve and print the name associated with the UID**

**if (Firebase.RTDB.get(&fbdo, path))**

**{**

**Serial.print("\n NAME: " + String(fbdo.stringData())); // Debug output**

**if (fbdo.dataType() == "string")**

**{**

**const char\* name = fbdo.stringData().c\_str(); // Use stringData() for string retrieval**

**Serial.println("\n Name for " + String(uid) + ": " + String(name)); // Print name on a separate line**

**}**

**else**

**{**

**Serial.println("\n Name data is not a string for " + String(uid));**

**}**

**}**

**else**

**{**

**Serial.println("Error getting name for " + String(uid) + ": " + fbdo.errorReason());**

**}**

**}**

**void printpin(const char\* path, const char\* uid)**

**{**

**// Retrieve and print the amount associated with the UID**

**if (Firebase.RTDB.getInt(&fbdo, path))**

**{**

**int pin = fbdo.intData(); // Assign amount from Firebase**

**Serial.println("\n pin for " + String(uid) + ": " + String(pin)); // Print amount on a separate line**

**}**

**else**

**{**

**Serial.println("Error getting pin for " + String(uid) + ": " + fbdo.errorReason());**

**}**

**}**

**OUTPUT:**

**20:56:06.196 -> Scanned UID: 3351e534**

**20:56:06.196 -> User Found: 3351e534**

**20:56:09.098 -> Amount for 3351e534: 5000**

**20:56:09.702 ->**

**20:56:09.702 -> NAME: Tharani**

**20:56:09.702 -> Name for 3351e534:**

**20:56:10.086 ->**

**20:56:10.086 -> pin for 3351e534: 1234**

**LCD AND KEYPAD INTERFACING :**

#include <pic.h>

// Configuration bits

\_\_CONFIG(FOSC\_HS & WDTE\_OFF & PWRTE\_OFF & CP\_OFF & BOREN\_ON & LVP\_OFF & CPD\_OFF & WRT\_OFF & DEBUG\_OFF);

#define \_XTAL\_FREQ 20000000 // Crystal frequency (20MHz)

// Define control pins for the LCD

#define rs RD2 // Register select

#define en RD3 // Enable

#define rw GND // Ground pin

// Define pins for the keypad

#define R1 RB0

#define R2 RB1

#define R3 RB2

#define R4 RB3

#define C1 RB4

#define C2 RB5

#define C3 RB6

#define C4 RB7

// Function prototypes

void lcd\_init();

void cmd(unsigned char a);

void dat(unsigned char b);

void show(const unsigned char \*s);

void lcd\_delay();

void clear\_input\_str(char \*str, unsigned char size);

void int\_to\_str(unsigned int value, char \*str);

unsigned char key();

void keyinit();

void power\_by();

void scan\_RFID();

void welcome();

void purchase();

void balance();

void total();

void pay\_page();

void pin();

void pay\_done();

void pay\_fail();

void thank\_you();

unsigned char keypad[4][4] = {

{'1', '2', '3', 'H'},

{'4', '5', '6', 'P'},

{'7', '8', '9', 'S'},

{'C', '0', '=', '+'}

};

// Global variables

unsigned char rowloc, colloc;

unsigned int total\_amount = 0;

void main() {

unsigned char b;

TRISD = 0; // Set PORTD as output for LCD

TRISB = 0xF0; // Set upper nibble as input for keypad

lcd\_init();

keyinit();

while (1) {

b = key();

if (b == 'C') {

cmd(0x01); // Clear the display

continue; // Skip the rest of the loop to avoid extra delays

}

if (b == '1') {

power\_by();

}

if (b == '2') {

scan\_RFID();

}

if (b == '3') {

welcome();

}

if (b == '4') {

purchase();

}

if (b == '5') {

balance();

}

if (b == '6') {

total();

}

if (b == '7') {

pay\_page();

}

if (b == '8') {

pin();

}

if (b == '9') {

pay\_done();

}

if (b == '0') {

pay\_fail();

}

if (b == 'H') {

thank\_you();

}

\_\_delay\_ms(100); // Added a delay for button debounce

}

}

void power\_by() {

cmd(0x01); // Clear display

cmd(0x81); // Move cursor to the first line

show("BYTES IN BITS");

cmd(0xC2);

show("Smart RFID");

cmd(0x91);

show("Payment system");

cmd(0xD0);

show("-Fire developers");

}

void scan\_RFID() {

cmd(0x01);

cmd(0x85);

show("Scan your");

cmd(0xc2);

show("RFID card");

cmd(0x90);

show("\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

cmd(0xD0);

show("-------------->");

}

void welcome() {

cmd(0x01);

cmd(0x80);

show("WELCOME");

cmd(0xC0);

show("1.Purchase:");

cmd(0x90);

show("2.Balance");

cmd(0xD0);

show("3.Exit");

\_\_delay\_ms(1000);

}

void purchase() {

cmd(0x01);

cmd(0x80);

show("Enter Amount:");

unsigned char digits[5] = " "; // Buffer for the display

unsigned char count = 0;

unsigned char b;

while (1) {

b = key();

if (b >= '0' && b <= '9' && count < 4) {

digits[count++] = b; // Store the digit

dat(b); // Display the digit

} else if (b == '+' || b == '-') {

if (count > 0) {

// Convert entered digits to an integer

unsigned int entered\_value = 0;

for (unsigned char i = 0; i < count; i++) {

entered\_value = entered\_value \* 10 + (digits[i] - '0');

}

if (b == '+') {

total\_amount += entered\_value;

} else if (b == '-') {

total\_amount -= entered\_value;

}

}

count = 0; // Reset count after operation

clear\_input\_str(digits, sizeof(digits)); // Clear digits after operation

} else if (b == '=') {

// Display total amount

cmd(0x01); // Clear display

cmd(0x80);

show("Total Amount:");

cmd(0xC0);

char total\_str[6]; // Enough space for total and null terminator

int\_to\_str(total\_amount, total\_str);

show(total\_str); // Display total amount

lcd\_delay(); // Short delay

cmd(0x80); // Move cursor back to the top

\_\_delay\_ms(1000);

break; // Exit the purchase mode

}

}

}

void balance() {

cmd(0x01);

cmd(0x80);

show("User balance:");

// Here you can display the actual balance if you have that variable

\_\_delay\_ms(1000);

}

void total() {

cmd(0x01);

cmd(0x80);

show("Total amount:");

char total\_str[6];

int\_to\_str(total\_amount, total\_str);

cmd(0xC0);

show(total\_str);

\_\_delay\_ms(1000);

}

void pay\_page() {

cmd(0x01);

cmd(0x80);

show("Payment page");

cmd(0xc0);

show("tharani");

\_\_delay\_ms(1000);

}

void pin() {

cmd(0x01);

cmd(0x80);

show("PIN:");

cmd(0xc0);

show("1234");

\_\_delay\_ms(1000);

}

void pay\_done() {

cmd(0x01);

cmd(0x80);

show("Payment Done");

\_\_delay\_ms(1000);

}

void pay\_fail() {

cmd(0x01);

cmd(0x80);

show("Payment Fail");

\_\_delay\_ms(1000);

}

void thank\_you() {

cmd(0x01);

cmd(0x80);

show("Thank You");

cmd(0xc0);

show("Come Again");

\_\_delay\_ms(1000);

}

void lcd\_init() {

cmd(0x02); // Return home

cmd(0x28); // 4-bit mode, 2 lines, 5x7 dots

cmd(0x0E); // Display on, cursor blinking

cmd(0x06); // Increment cursor

cmd(0x80); // Move cursor to the first line

lcd\_delay(); // Added delay after initialization

}

void cmd(unsigned char a) {

rs = 0;

PORTD &= 0x0F;

PORTD |= (a & 0xf0);

en = 1;

lcd\_delay();

en = 0;

lcd\_delay();

PORTD &= 0x0F;

PORTD |= (a << 4 & 0xf0);

en = 1;

lcd\_delay();

en = 0;

lcd\_delay();

}

void dat(unsigned char b) {

rs = 1;

PORTD &= 0x0F;

PORTD |= (b & 0xf0);

en = 1;

lcd\_delay();

en = 0;

lcd\_delay();

PORTD &= 0x0F;

PORTD |= (b << 4 & 0xf0);

en = 1;

lcd\_delay();

en = 0;

lcd\_delay();

}

void show(const unsigned char \*s) {

while (\*s) {

dat(\*s++);

}

}

void lcd\_delay() {

unsigned int lcd\_delay;

for (lcd\_delay = 0; lcd\_delay <= 1000; lcd\_delay++);

}

void clear\_input\_str(char \*str, unsigned char size) {

for (unsigned cn.har i = 0; i < size; i++) {

str[i] = ' '; // Fill with spaces

}

str[size] = '\0'; // Null-terminate the string

}

void int\_to\_str(unsigned int value, char \*str) {

unsigned char i = 0;

if (value == 0) {

str[i++] = '0';

} else {

while (value > 0) {

str[i++] = (value % 10) + '0'; // Get the last digit

value /= 10; // Remove the last digit

}

}

str[i] = '\0'; // Null-terminate the string

// Reverse the string

for (unsigned char j = 0; j < i / 2; j++) {

char temp = str[j];

str[j] = str[i - j - 1];

str[i - j - 1] = temp;

}

}

void keyinit() {

OPTION\_REG |= 0x7F; // Enable pull-ups

}

unsigned char key() {

PORTB = 0x00; // Clear PORTB

while (C1 && C2 && C3 && C4); // Wait for key press

while (!C1 || !C2 || !C3 || !C4) {

R1 = 0; R2 = R3 = R4 = 1;

if (!C1 || !C2 || !C3 || !C4) {

rowloc = 0;

break;

}

R2 = 0; R1 = 1;

if (!C1 || !C2 || !C3 || !C4) {

rowloc = 1;

break;

}

R3 = 0; R2 = 1;

if (!C1 || !C2 || !C3 || !C4) {

rowloc = 2;

break;

}

R4 = 0; R3 = 1;

if (!C1 || !C2 || !C3 || !C4) {

rowloc = 3;

break;

}

}

if (C1 == 0 && C2 != 0 && C3 != 0 && C4 != 0)

colloc = 0;

else if (C1 != 0 && C2 == 0 && C3 != 0 && C4 != 0)

colloc = 1;

else if (C1 != 0 && C2 != 0 && C3 == 0 && C4 != 0)

colloc = 2;

else if (C1 != 0 && C2 != 0 && C3 != 0 && C4 == 0)

colloc = 3;

while (C1 == 0 || C2 == 0 || C3 == 0 || C4 == 0);

return (keypad[rowloc][colloc]);

}

18/10/2024

Fetch the Name, Amount and Pin Display in LCD

Code:

Pic:

#include <pic.h>

#include <stdio.h>

// Configuration bits

\_\_CONFIG(FOSC\_HS & WDTE\_OFF & PWRTE\_OFF & CP\_OFF & BOREN\_ON & LVP\_OFF & CPD\_OFF & WRT\_OFF & DEBUG\_OFF);

#define \_XTAL\_FREQ 20000000 // Crystal frequency (20MHz)

// Define control pins for the LCD

#define rs RD2 // Register select

#define en RD3 // Enable

#define rw GND // Ground pin

// Function prototypes

void lcd\_init();

void cmd(unsigned char a);

void dat(unsigned char b);

void show(const unsigned char \*s);

void uart\_init(void);

char uart\_receive(void);

void uart\_receive\_string(char \*buffer, unsigned char size);

void lcd\_delay();

// Global variable

char received\_string[64]; // Buffer to store the received string (increased size)

void main() {

// Port initialization

TRISD = 0; // Set PORTD as output for LCD

uart\_init();

lcd\_init();

while (1) {

// Receive a string from the UART

uart\_receive\_string(received\_string, sizeof(received\_string));

cmd(0x01); // Clear display

// Custom parsing of the received string

char name[16], amount[16], pin[16];

int i = 0, j = 0;

// Extract name

while (received\_string[i] != ',' && received\_string[i] != '\0') {

name[j++] = received\_string[i++];

}

name[j] = '\0'; // Null-terminate the name

i++; // Move past the comma

// Extract amount

j = 0;

while (received\_string[i] != ',' && received\_string[i] != '\0') {

amount[j++] = received\_string[i++];

}

amount[j] = '\0'; // Null-terminate the amount

i++; // Move past the comma

// Extract pin

j = 0;

while (received\_string[i] != '\0' && received\_string[i] != '\n' && received\_string[i] != '\r') {

pin[j++] = received\_string[i++];

}

pin[j] = '\0'; // Null-terminate the pin

// Display name on the first line

cmd(0x80); // Move cursor to the first line

show(name); // Display name

// Display amount on the second line

cmd(0xC0); // Move cursor to the second line

show(amount); // Display amount

// Display pin on the third line

cmd(0x94); // Move cursor to the third line

show(pin); // Display pin

\_\_delay\_ms(2000); // Delay to see the messages

}

}

void lcd\_init() {

cmd(0x02); // Return home

cmd(0x28); // 4-bit mode, 2 lines, 5x7 dots

cmd(0x0E); // Display on, cursor blinking

cmd(0x06); // Increment cursor

cmd(0x80); // Move cursor to the first line

lcd\_delay(); // Added delay after initialization

}

void cmd(unsigned char a) {

rs = 0; // Command mode

PORTD &= 0x0F; // Clear upper nibble

PORTD |= (a & 0xF0); // Send high nibble

en = 1; // Enable pulse

lcd\_delay();

en = 0;

PORTD &= 0x0F; // Clear upper nibble

PORTD |= (a << 4); // Send low nibble

en = 1; // Enable pulse

lcd\_delay();

en = 0;

}

void dat(unsigned char b) {

rs = 1; // Data mode

PORTD &= 0x0F; // Clear upper nibble

PORTD |= (b & 0xF0); // Send high nibble

en = 1; // Enable pulse

lcd\_delay();

en = 0;

PORTD &= 0x0F; // Clear upper nibble

PORTD |= (b << 4); // Send low nibble

en = 1; // Enable pulse

lcd\_delay();

en = 0;

}

void show(const unsigned char \*s) {

unsigned char line = 0; // Track the current line

unsigned char count = 0; // Track character count on the current line

while (\*s) {

// If we reach the end of a line (16 characters), move to the next line

if (count >= 16) {

count = 0;

line++;

// Move to the start of the next line, if within bounds

if (line < 4) {

cmd(0xC0 + (line \* 0x40)); // Move to line 2, 3, or 4

} else {

// If we exceed 4 lines, stop processing

break;

}

}

// Send the character to the LCD

dat(\*s++);

count++; // Increment the character count

}

}

void lcd\_delay() {

unsigned int lcd\_delay;

for (lcd\_delay = 0; lcd\_delay < 1000; lcd\_delay++);

}

void uart\_init(void) {

TRISC6 = 0; // TX pin set as output

TRISC7 = 1; // RX pin set as input

SPBRG = 129; // Baud rate 9600 for 20MHz

BRGH = 1; // High baud rate

SYNC = 0; // Asynchronous mode

SPEN = 1; // Enable serial port

TXEN = 1; // Enable transmission

CREN = 1; // Enable continuous reception

TX9 = 0; // 8-bit transmission

RX9 = 0; // 8-bit reception

}

char uart\_receive(void) {

while (!RCIF); // Wait for reception to complete

return RCREG; // Return received data

}

void uart\_receive\_string(char \*buffer, unsigned char size) {

unsigned char i = 0;

char ch;

while (i < size - 1) { // Leave space for null terminator

ch = uart\_receive(); // Receive one character

if (ch == '\n' || ch == '\r') { // Check for newline or carriage return

buffer[i] = '\0'; // Null-terminate the string

break; // Exit loop if end of string

}

buffer[i++] = ch; // Store character in buffer

}

buffer[i] = '\0'; // Null-terminate the string

}

Ardiuno:

#include <SPI.h>

#include <MFRC522.h>

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <Firebase\_ESP\_Client.h>

#include <ArduinoJson.h>

#include <addons/TokenHelper.h>

#include <addons/RTDBHelper.h>

#include <SoftwareSerial.h>

// RFID Setup

#define SS\_PIN 2 // SDA

#define RST\_PIN 0 // RST

MFRC522 rfid(SS\_PIN, RST\_PIN);

// WiFi and Firebase Setup

#define WIFI\_SS "Galaxy A10s5584"

#define WIFI\_PASSWORD "isaitharani"

#define API\_KEY "AIzaSyCi-2z5tjB8LLBbMCZeuewGi6tSQtF9iPg"

#define DATABASE\_URL "https://smart-way-payment-default-rtdb.firebaseio.com"

#define USER\_EMAIL "thirisha1227@gmail.com"

#define USER\_PASSWORD "Thirisha@2711"

FirebaseData fbdo;

FirebaseAuth auth;

FirebaseConfig config;

// Updated UIDs for lookup

const char\* uids[] = {"af14e91c", "3351e534", "8f13ea1f"}; // Add your UIDs for lookup

// Define RX and TX pins for communication with PIC

#define RX\_PIN 3

#define TX\_PIN 1

SoftwareSerial mySerial(RX\_PIN, TX\_PIN); // RX, TX

void setup() {

Serial.begin(115200);

mySerial.begin(9600); // Communication with PIC

// Initialize RFID

SPI.begin();

rfid.PCD\_Init();

Serial.println("Scan an RFID tag...");

// Connect to Wi-Fi

WiFi.begin(WIFI\_SS, WIFI\_PASSWORD);

Serial.print("Connecting to Wi-Fi");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(300);

}

Serial.println();

Serial.print("Connected with IP: ");

Serial.println(WiFi.localIP());

Serial.println();

// Set up Firebase configuration

config.api\_key = API\_KEY;

auth.user.email = USER\_EMAIL;

auth.user.password = USER\_PASSWORD;

config.database\_url = DATABASE\_URL;

Firebase.begin(&config, &auth);

Firebase.reconnectWiFi(true);

}

void loop() {

// Check for RFID card presence

if (!rfid.PICC\_IsNewCardPresent()) {

return;

}

// Read RFID card

if (!rfid.PICC\_ReadCardSerial()) {

return;

}

// Get UID as a string

String scannedUID = "";

for (byte i = 0; i < rfid.uid.size; i++) {

scannedUID += (rfid.uid.uidByte[i] < 0x10 ? "0" : "") + String(rfid.uid.uidByte[i], HEX);

}

Serial.println("Scanned UID: " + scannedUID);

// Match the scanned UID with predefined UIDs

for (int i = 0; i < sizeof(uids) / sizeof(uids[0]); i++) {

if (scannedUID.equalsIgnoreCase(uids[i])) {

Serial.println("User Found: " + String(uids[i]));

fetchData(uids[i]); // Fetch data for the matched UID

break;

}

}

// Halt PICC

rfid.PICC\_HaltA();

}

void fetchData(const char\* uid) {

// Set the paths for amount, name, and pin

String pathAmount = "UID/" + String(uid) + "/amount";

String pathName = "UID/" + String(uid) + "/name";

String pathPin = "UID/" + String(uid) + "/pin";

// Fetch and print the amount, name, and pin

String name = fetchName(pathName.c\_str(), uid);

int amount = fetchAmount(pathAmount.c\_str(), uid);

int pin = fetchPin(pathPin.c\_str(), uid);

// Send formatted data to PIC

mySerial.print(String(name) + "," + String(amount) + "," + String(pin) + "\n");

}

String fetchName(const char\* path, const char\* uid) {

// Retrieve and return the name associated with the UID

if (Firebase.RTDB.get(&fbdo, path)) {

if (fbdo.dataType() == "string") {

return fbdo.stringData().c\_str(); // Use stringData() for string retrieval

}

} else {

Serial.println("Error getting name for " + String(uid) + ": " + fbdo.errorReason());

}

return "Unknown"; // Default if not found

}

int fetchAmount(const char\* path, const char\* uid) {

// Retrieve and return the amount associated with the UID

if (Firebase.RTDB.getInt(&fbdo, path)) {

return fbdo.intData(); // Return amount from Firebase

} else {

Serial.println("Error getting amount for " + String(uid) + ": " + fbdo.errorReason());

}

return 0; // Default if not found

}

int fetchPin(const char\* path, const char\* uid) {

// Retrieve and return the pin associated with the UID

if (Firebase.RTDB.getInt(&fbdo, path)) {

return fbdo.intData(); // Return pin from Firebase

} else {

Serial.println("Error getting pin for " + String(uid) + ": " + fbdo.errorReason());

}

return 0; // Default if not found

}`