Your Partner for Excellence

Texas College of Management & IT

Texas College of Management and IT

IOT Project Report

Explore, Learn and Excel

Professional Class

Submitted By: Mohammad Rahish Shekh

Submitted To: Department Of IT

LCID: LC00017002687

Program: BIT

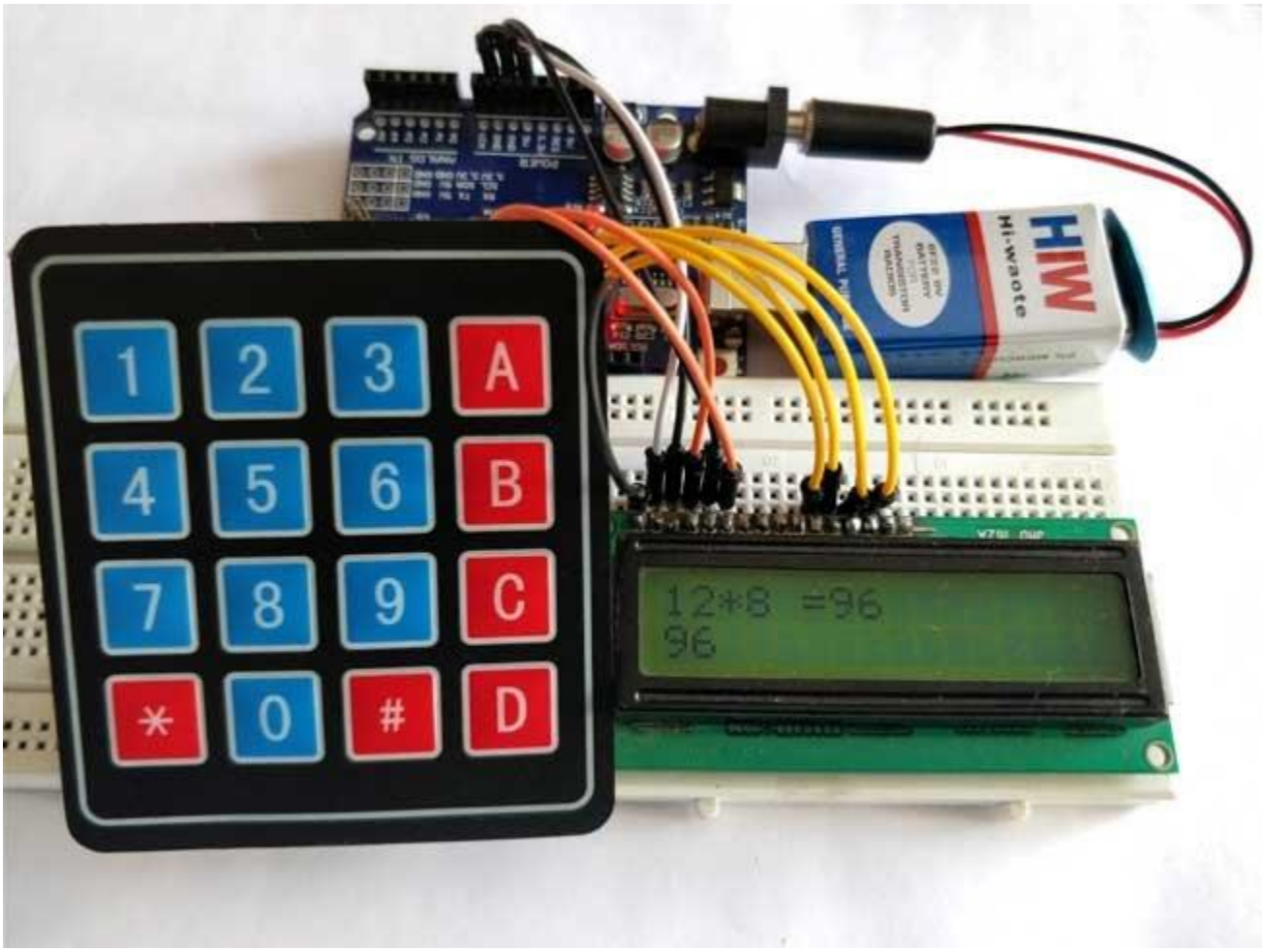
Sections: F

Date:25 June

IOT And Robotics Report

Smart Calculator

"Speak and Compute: Calculations at Your Command"



Abstract

This project presents the development of a versatile and innovative calculator using an Arduino Uno, a 2x16 LCD, and a 4x4 keypad. The primary objective was to create a user-friendly and efficient calculator with enhanced interactivity through voice recognition, enabling control via voice commands and manual keypad input. The integration of voice recognition technology adds a smart functionality, allowing users to perform calculations hands-free by speaking commands into a connected microphone. Combined with traditional keypad input, this feature offers flexibility and ease of use, making the calculator accessible to a broader range of users. The implementation involves programming the Arduino Uno to interpret both keypad entries and voice commands, process the input, and display the results on the LCD screen. This project showcases the practical application of microcontroller programming and highlights the potential of combining hardware components and software algorithms to create intelligent and interactive systems.

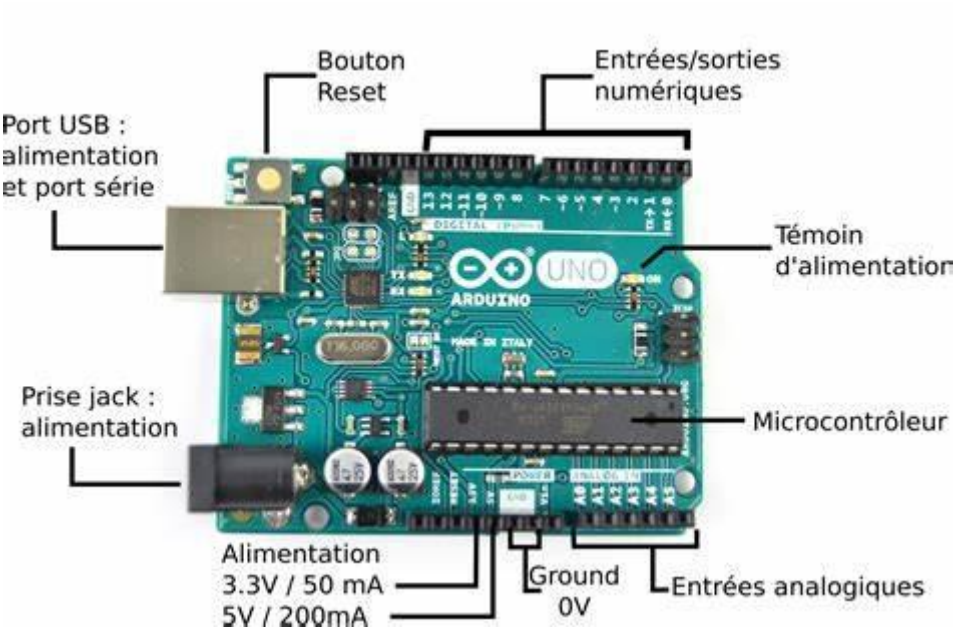
Table of Contents

1. Introduction

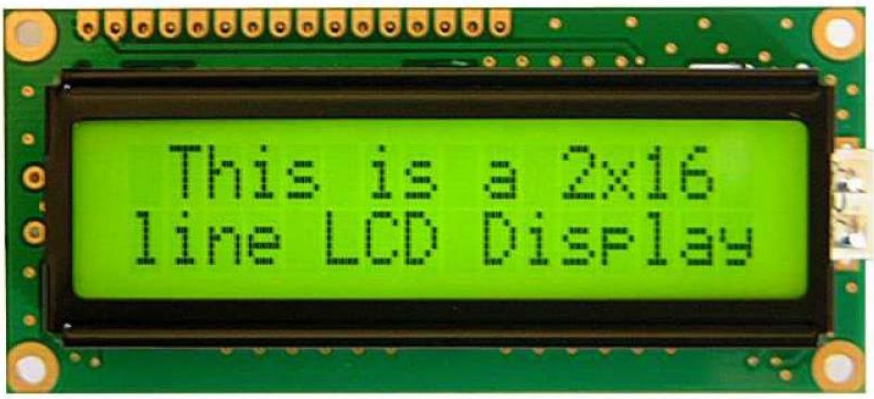
The purpose of this project is to design and implement a smart calculator that can be controlled both by voice commands and a keypad. This system utilizes an Arduino Uno microcontroller, a 2x16 LCD display for output, and a 4x4 matrix keypad for manual input. Additionally, the calculator incorporates a voice recognition module to interpret and process spoken commands, thereby enhancing user interactivity and accessibility.

2. Components

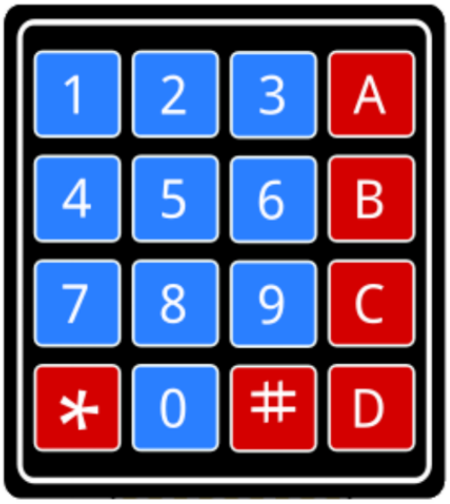
Arduino Uno: The microcontroller serves as the brain of the calculator, handling all input processing, calculations, and output.



2x16 LCD Display: This component is used to display inputs, results, and feedback to the user.



4x4 Matrix Keypad: The keypad allows users to manually input numbers and operations.



Python Code for Serial Communication: This code runs on a connected computer to process voice commands captured via the computer's microphone and communicate the commands to the Arduino.

3. System Design

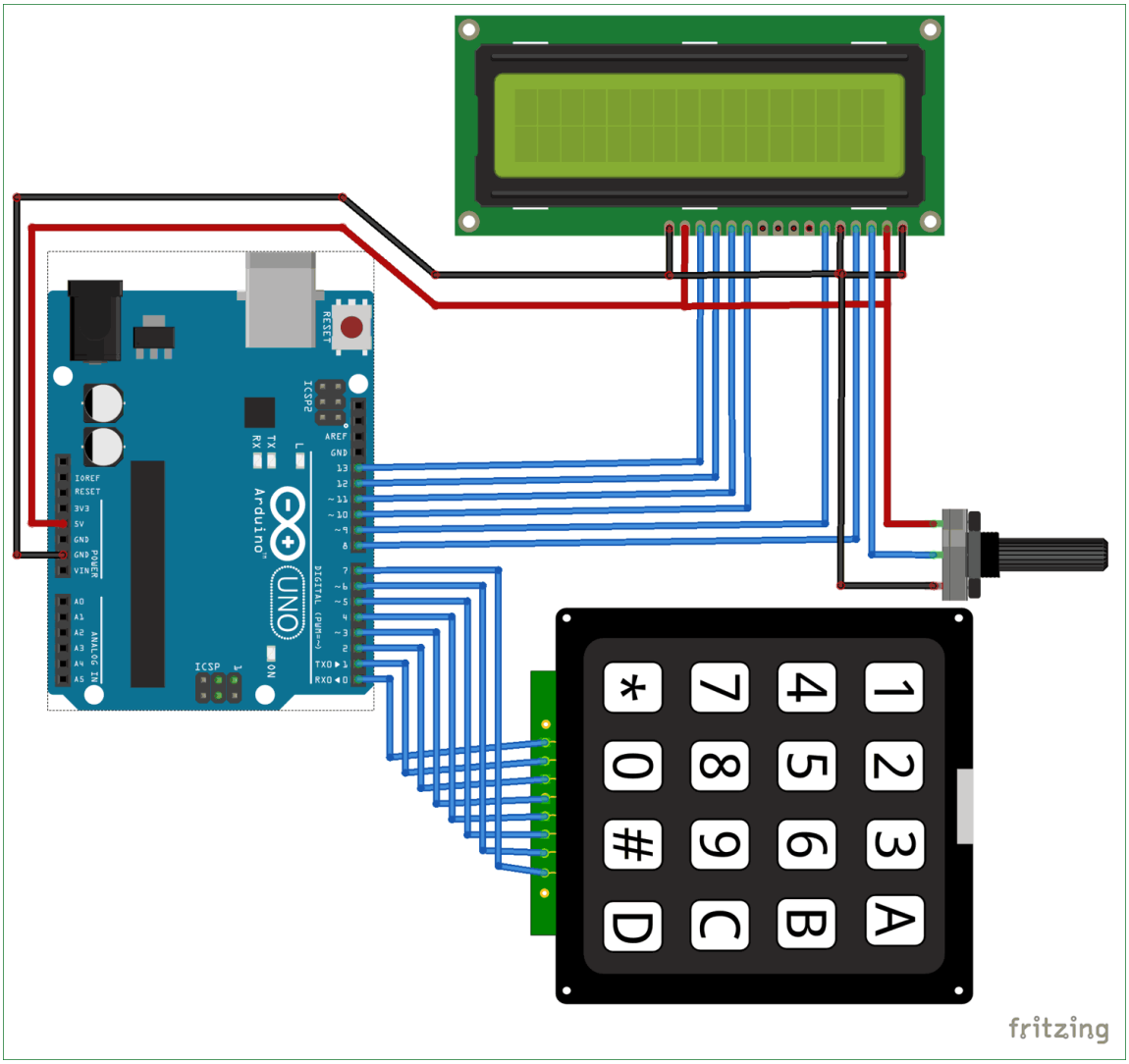
Hardware Integration:

1. The Arduino Uno is connected to the 2x16 LCD display using the appropriate pins for power, ground, and data transfer.

2. The **4x4 keypad** is connected to the **Arduino** through digital I/O pins, allowing it to read button presses.

3. The computer running Python code for voice recognition communicates with the Arduino via a serial connection.

Circuit Diagram:



4. Software Development:

Keypad Input Handling: The Arduino is programmed to read inputs from the keypad, detect which button is pressed, and perform the corresponding action.

Voice Command Processing: The Python script captures voice commands using the computer's microphone, processes these commands, and sends the corresponding data to the Arduino over a serial connection. The Arduino interprets these commands to execute the appropriate operations.
Calculation Logic: The core logic for performing arithmetic operations (addition, subtraction, multiplication, division) is implemented in the Arduino code. It processes inputs from both the keypad and voice commands.

Display Management: The Arduino controls the LCD display to show inputs, intermediate steps, and final results.

5. Implementation

Setup and Initialization:

1. Initialize the LCD display, setting the cursor position and clearing the screen.

2. Configure the keypad matrix and set up the pins for reading inputs.

3. Set up the serial communication between the Arduino and the computer running the Python script.

4. Initialize the Python script for capturing and processing voice commands.

6. Operational Workflow:

Keypad Input: When a button is pressed on the keypad, the Arduino reads the input, updates the display, and stores the value for processing.

Voice Input: The Python script captures voice commands via the computer's microphone, processes them, and sends the corresponding data to the Arduino over the serial connection. The Arduino then performs the instructed operation.

Calculation and Display: The Arduino processes the inputs (from either the keypad or voice commands), performs the calculation, and updates the LCD display with the result.

7. Conclusion

The smart calculator project demonstrates the effective integration of hardware components and software algorithms to create a versatile and user-friendly device. By combining traditional keypad input with advanced voice recognition capabilities, the calculator offers enhanced accessibility and interactivity. This project serves as a valuable educational tool, showcasing the potential of microcontroller-based systems in creating intelligent and interactive devices.

8. Appendices

Source Code:

1. Python Script for serial communication:

```
import serial
import speech_recognition as sr
import pyttsx3
import os

# Initialize serial communication with Arduino
```

```
arduino = serial.Serial('COM3', 9600)  # Adjust COM port as needed

# Initialize the speech recognizer and text-to-speech engine
recognizer = sr.Recognizer()
engine = pyttsx3.init()

# Function to send commands to the Arduino
def send_command(command):
    arduino.write(command.encode())

# Function to speak out the results or commands
def speak(text):
    engine.say(text)
    engine.runAndWait()

# Function to recognize speech and interpret it as a command
def takeCommand():
    with sr.Microphone() as source:
        print("Listening...")
        recognizer.pause_threshold = 1
        audio = recognizer.listen(source)

    try:
        print("Recognizing...")
        query = recognizer.recognize_google(audio, language='en-in')
        print("User said:", query)
    except Exception as e:
        print("Say that again please...")
        return "None"
    return query

# Function to sign up a new user
def sign_up():
    speak("Please say your new ID")
    new_id = takeCommand().lower()
    speak("Please say your new password")
    new_password = takeCommand().lower()

    with open("authentication.txt", "r") as auth_file:
        auth_data = auth_file.readlines()
        for line in auth_data:
            stored_data = line.strip().split(maxsplit=1)
            if len(stored_data) >= 2 and stored_data[0] == new_id:
                speak("This ID already exists. Please choose a different ID.")
                return
```

```
    with open("authentication.txt", "a") as auth_file:
        auth_file.write(f"{new_id} {new_password}\n")

    speak("Sign up successful. You can now log in.")

# Function to authenticate user
def authenticate():
    speak("Please say your ID")
    user_id = takeCommand().lower()
    speak("Please say your password")
    user_password = takeCommand().lower()

    with open("authentication.txt", "r") as auth_file:
        auth_data = auth_file.readlines()

    for line in auth_data:
        stored_data = line.strip().split(maxsplit=1)
        if len(stored_data) >= 2:
            stored_id, stored_password = stored_data[0], stored_data[1]
            if stored_id == user_id:
                if stored_password == user_password:
                    speak("Access granted. Welcome to the Smart Calculator System
designed by Raahis Sheikh.")
                    return True
                else:
                    speak("The password you entered is incorrect.")
                    return False

    speak("This ID does not exist. Please sign up first.")
    return False

# Function to check password by ID
def check_password_by_id():
    speak("Please say your ID")
    user_id = takeCommand().lower()

    with open("authentication.txt", "r") as auth_file:
        auth_data = auth_file.readlines()

    for line in auth_data:
        stored_data = line.strip().split(maxsplit=1)
        if len(stored_data) >= 2:
            stored_id, stored_password = stored_data[0], stored_data[1]
            if stored_id == user_id:
```

```
                speak(f"The password for ID {user_id} is {stored_password}")
                print(f"The password for ID {user_id} is {stored_password}")
                return

    speak("This ID does not exist.")

# Function to process recognized commands
def process_command(command):
    def send_number(num):
        send_command(str(num))
        speak(str(num))

    # Basic arithmetic commands
    if "write one" in command or "right one" in command or "right van" in command
or "1" in command:
        send_command('1')
        speak("One")
    elif "write two" in command or "right to" in command or "right tu" in
command:
        send_command('2')
        speak("Two")
    elif "write three" in command or "right three" in command or "three" in
command:
        send_command('3')
        speak("Three")
    elif "write four" in command:
        send_command('4')
        speak("Four")
    elif "write five" in command:
        send_command('5')
        speak("Five")
    elif "write six" in command:
        send_command('6')
        speak("Six")
    elif "write seven" in command:
        send_command('7')
        speak("Seven")
    elif "write eight" in command:
        send_command('8')
        speak("Eight")
    elif "write nine" in command:
        send_command('9')
        speak("Nine")
    elif "write zero" in command:
        send_command('0')
```

```
        speak("Zero")
    elif "write plus" in command or "right plus" in command:
        send_command('A')
        speak("Plus")
    elif "write minus" in command or "subtract" in command:
        send_command('B')
        speak("Minus")
    elif "write multiply" in command or "times" in command:
        send_command('C')
        speak("Multiply")
    elif "write divide" in command or "divide by" in command:
        send_command('D')
        speak("Divide")
    elif "clear the display" in command or "clear display" in command:
        send_command('*')
        speak("Display cleared")
    elif "write equals" in command or "right equals" in command:
        send_command('#')
        speak("Equals")

    # Extra communication jarvis
    elif "hi" in command:
        speak("Hello, how can I help you?")

    elif "hello" in command and "jarvis" in command:
        speak("Jarvis wasn't died in iron man movies")
        speak("I am still here")
        speak("Hello there")

    elif "can you help me" and "calculations" in command:
        speak("Yes, I can help you with some arithmetic calculations")
        speak("I can send the command you speak to the calculator connected to
me")
        speak("and the result will be seen on the display")

    # Addition, subtraction, multiplication, and division commands
    elif "add" in command or "plus" in command or "find the sum" in command:
        numbers = [int(s) for s in command.split() if s.isdigit()]
        if len(numbers) == 2:
            send_number(numbers[0])
            send_command('A')
            speak("plus")
            send_number(numbers[1])
            send_command('#')
            speak("equals")
```

```
            speak("The result has been updated on the LCD screen.")
        else:
            speak("Please provide two numbers for addition.")
    elif "subtract" in command or "find the difference" in command:
        numbers = [int(s) for s in command.split() if s.isdigit()]
        if len(numbers) == 2:
            send_number(numbers[0])
            send_command('B')
            speak("minus")
            send_number(numbers[1])
            send_command('#')
            speak("equals")
            speak("The result has been updated on the LCD screen.")
        else:
            speak("Please provide two numbers for subtraction.")
    elif "multiply" in command or "times" in command or "find the product" in
command:
        numbers = [int(s) for s in command.split() if s.isdigit()]
        if len(numbers) == 2:
            send_number(numbers[0])
            send_command('C')
            speak("multiply")
            send_number(numbers[1])
            send_command('#')
            speak("equals")
            speak("The result has been updated on the LCD screen.")
        else:
            speak("Please provide two numbers for multiplication.")
    elif "divide" in command or "find the division" in command:
        numbers = [int(s) for s in command.split() if s.isdigit()]
        if len(numbers) == 2:
            send_number(numbers[0])
            send_command('D')
            speak("divide")
            send_number(numbers[1])
            send_command('#')
            speak("equals")
            speak("The result has been updated on the LCD screen.")
        else:
            speak("Please provide two numbers for division.")
    else:
        speak("Command not recognized")

# Check if authentication file exists, if not create one
if not os.path.exists("authentication.txt"):
```

```
    open("authentication.txt", "w").close()

# Main loop to keep listening for commands
while True:
    speak("Do you want to sign up or log in?")
    action = takeCommand().lower()

    if "sign up" in action:
        sign_up()
    elif "login" in action:
        if authenticate():
            while True:
                command = takeCommand().lower()
                if command != "none":
                    process_command(command)

    elif "check my id" in action:
        check_password_by_id()
    elif "hello" in action:
        speak("Hello there, please signup or login to your account")
    else:
        speak("It seems I couldn't understand your command")
        speak("Or you have spoken wrong command?")
        speak("Please say 'sign up' or 'login' or 'check my id'")
```

2. Code for Arduino:

```
#include <LiquidCrystal.h>
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins[ROWS] = {0, 1, 2, 3};
byte colPins[COLS] = {4, 5, 6, 7};
```

```
Keypad kpd = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

long Num1, Num2, Number;
char key, action;
boolean result = false;

void setup() {
  lcd.begin(16, 2);
  lcd.print("DIY Calculator");
  lcd.setCursor(0, 1);
  lcd.print("-CircuitDigest");
  delay(2000);
  lcd.clear();
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    key = Serial.read();
    if (key != NO_KEY) {
      DetectButtons();
    }
  } else {
    key = kpd.getKey();
    if (key != NO_KEY) {
      DetectButtons();
    }
  }

  if (result == true) {
    CalculateResult();
  }

  DisplayResult();
}

void DetectButtons() {
  lcd.clear();
  if (key == '*') {
    Number = Num1 = Num2 = 0;
    result = false;
```

```
  }

  if (key >= '0' && key <= '9') {
    int digit = key - '0';
    if (Number == 0) {
      Number = digit;
    } else {
      Number = (Number * 10) + digit;
    }
  }

  if (key == '#') {
    Num2 = Number;
    result = true;
  }

  if (key == 'A' || key == 'B' || key == 'C' || key == 'D') {
    Num1 = Number;
    Number = 0;
    if (key == 'A') {
      action = '+';
    }
    if (key == 'B') {
      action = '-';
    }
    if (key == 'C') {
      action = '*';
    }
    if (key == 'D') {
      action = '/';
    }
    delay(100);
  }
}

void CalculateResult() {
  if (action == '+')
    Number = Num1 + Num2;
  if (action == '-')
    Number = Num1 - Num2;
  if (action == '*')
    Number = Num1 * Num2;
  if (action == '/')
    Number = Num1 / Num2;
}
```

```
void DisplayResult() {
  lcd.setCursor(0, 0);
  lcd.print(Num1);
  lcd.print(action);
  lcd.print(Num2);

  if (result == true) {
    lcd.print(" =");
    lcd.print(Number);
  }

  lcd.setCursor(0, 1);
  lcd.print(Number);
}
```

The End