Using the 4x4 Matrix Keypad with a BeagleBone Y-AI

April 2025

This guide will allow you to:

Hook up a 4x4 Matrix Keypad to your BeagleBone Y-AI using the Adafruit GPIO Expander as there are not enough GPIO pins accessible on the board. This is a tailored guide to the specific hardware mentioned.

Required Hardware

BeagleBone Y-AI with breadboard
Adafruit 4x4 Matrix Keypad (3844)
Adafruit GPIO Expander Bonnet (4132)
STEMMA QT / Qwiic JST SH 4-pin Cable with Premium Female Sockets
17x Male/Female Jumper Cables
8x 10K-ohm resistors

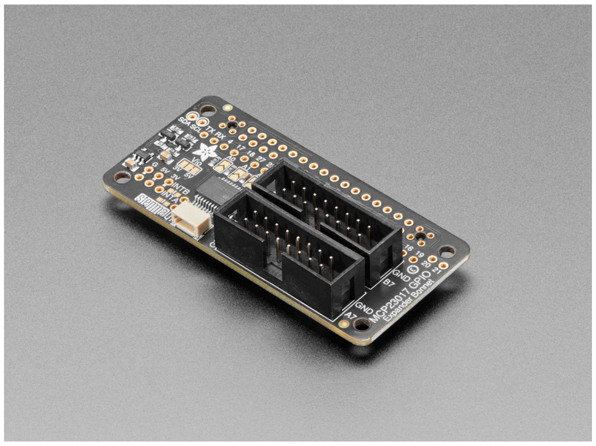
References

https://learn.adafruit.com/matrix-keypad/pinouts
https://www.adafruit.com/product/4397#technical-details
https://www.youtube.com/watch?v=7-PoXmxeCmQ
Dr. Brian’s lecture notes on Electronics

Keypad

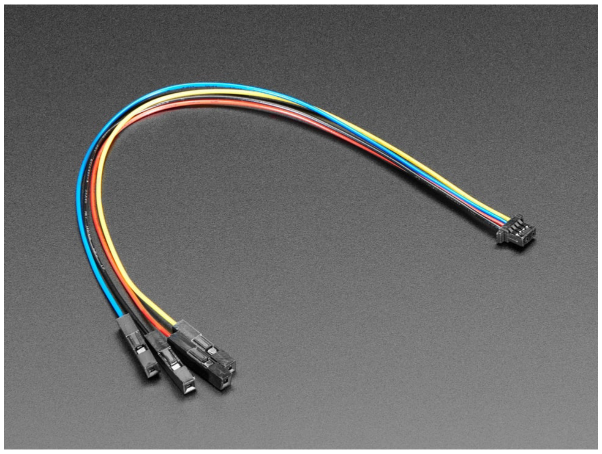
This matrix-connected keypad allows for 16 inputs that can be processed in an intuitive way.

GPIO Expander Bonnet



This GPIO Expander allows for 16 additional GPIO pins to connect various devices.

Expander Cable

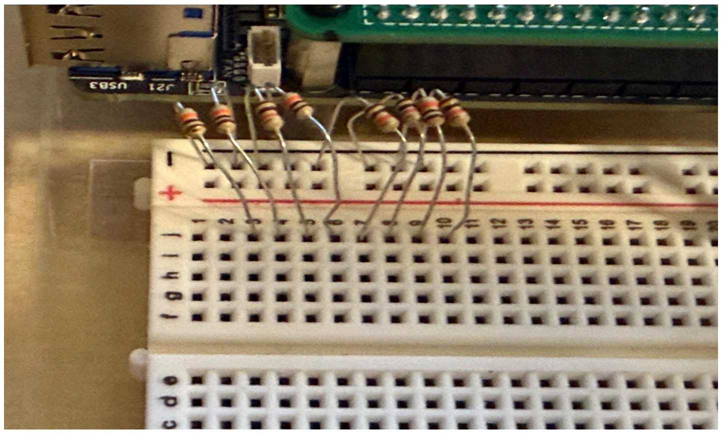


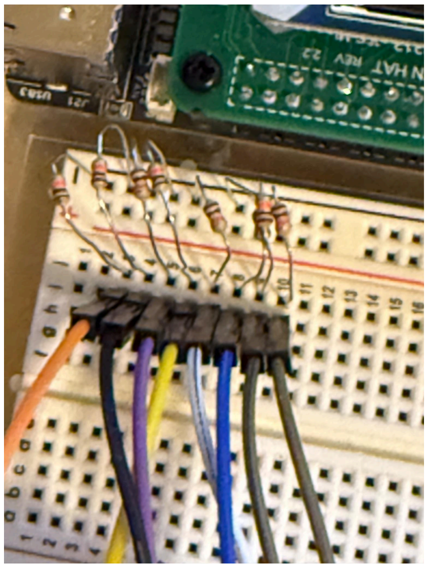
This cable allows for easy wiring of the GPIO Expander to the BeagleBone.

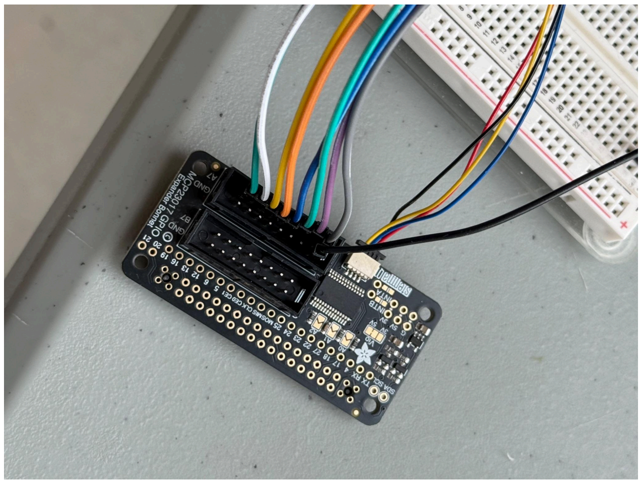
Wiring Steps

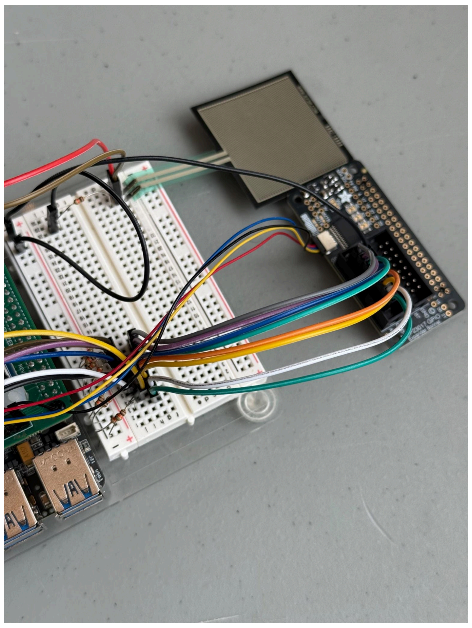
NOTE: START WITH THE BEAGLEBONE OFF, DO NOT TURN ON THE BEAGLEBONE WITHOUT PLACING/CONFIRMING ALL CIRCUITRY!

The wiring for the keypad and the GPIO Expander may clutter your board, consider using electrical tape to hold it in place and other measures for cable management.

1. Let's get the resistors set up on the breadboard first.
   Take your 8x 10K-ohm resistors and set them up so one end is hooked up to the negative (GND) row.
   

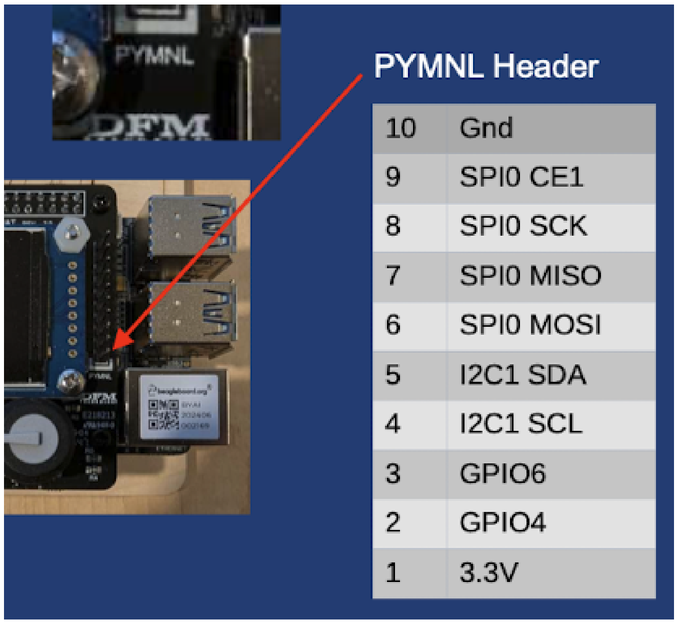
2. Now, let's hook up the keypad side.
   Take 8x of the jumper cables and connect them to the pins at the bottom of the keypad. Take each of the other ends of the cables and install in series with the resistors.
   

3. Now we can set up the GPIO Expander.
   The GPIO Expander has 2 sets of pins, refer to the image below. The bottom row of both sections are used as GND. We will use and refer to the pins in the top row of section A (A7) as pins 0-7.
   Hook up the 8x remaining, unconnected jumper cables to A7.
   

4. Next, take the other ends of the cables and connect them in series to the existing circuitry on the breadboard as shown below. (Note, one wire went missing in the image)
   

5. Next, let's connect the Expander to the BeagleBone. Connect the Expander Cable's JST-SH female 4-pin side to the Stemma QT port on the Expander. Refer to the image above for clarity. The other end of the Expander Cable will be connected to the PYMNL header.

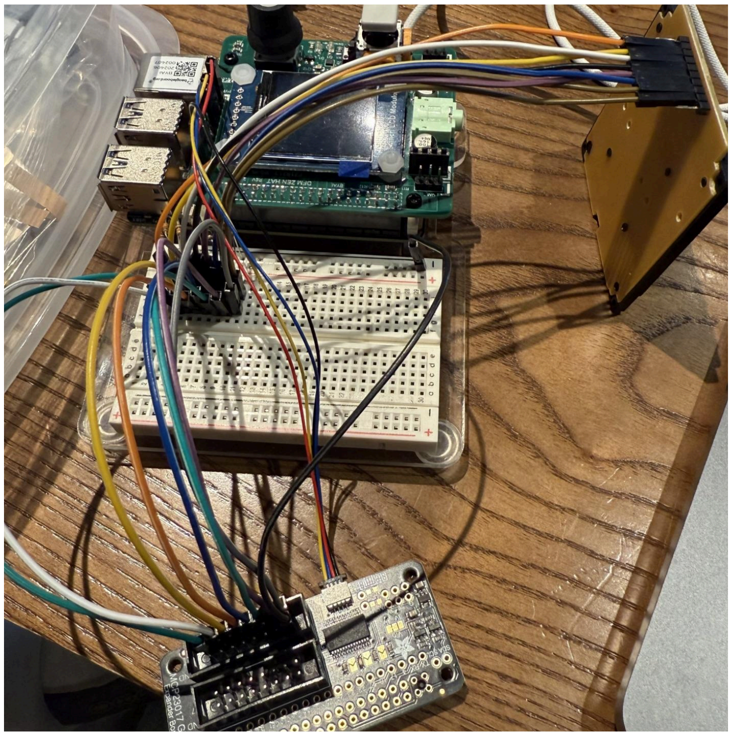
6. Connect the Expander Cable’s female black wire to pin 10 for GND.
   Connect the blue wire to pin 5 - I2C1 SDA Data.
   Connect the yellow wire to pin 4 for I2C1 SCL Clock.
   Connect the red wire to pin 1 for 3.3V power.



PYMNL Header

10	Gnd
9	SPI0 CE1
8	SPI0 SCK
7	SPI0 MISO
6	SPI0 MOSI
5	I2C1 SDA
4	I2C1 SCL
3	GPIO6
2	GPIO4
1	3.3V

7. Next, take the last jumper cable to connect the negative row of the breadboard to the one of the GND pins on the GPIO Expander.
   Your complete circuit should look as below.



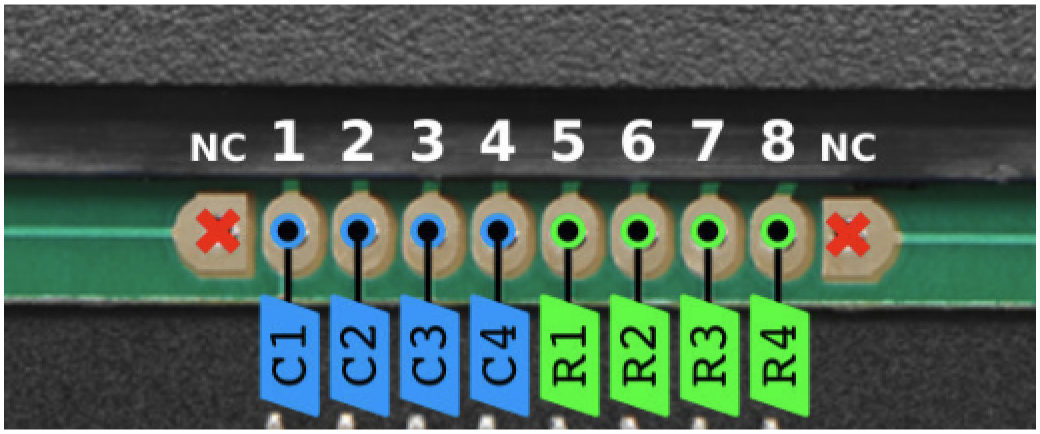
Reading Input

On Host, enter the following command:

```
(HOST) echo mcp23017 0x20 | sudo tee /sys/class/i2c-adapter/i2c-1/new_device
```

This will register the MCP23017 device (Expander) with the I2C1 bus at address 0x20 (use i2cdetect to check this) to instantiate the driver and expose it as GPIOCHIP3.
Now, you can use GPIOCHIP3 in your gpio.c code to toggle and read the pins.

This is the pin layout for the keypad. The first four pins are for the columns and the last four are for the rows of the keypad.



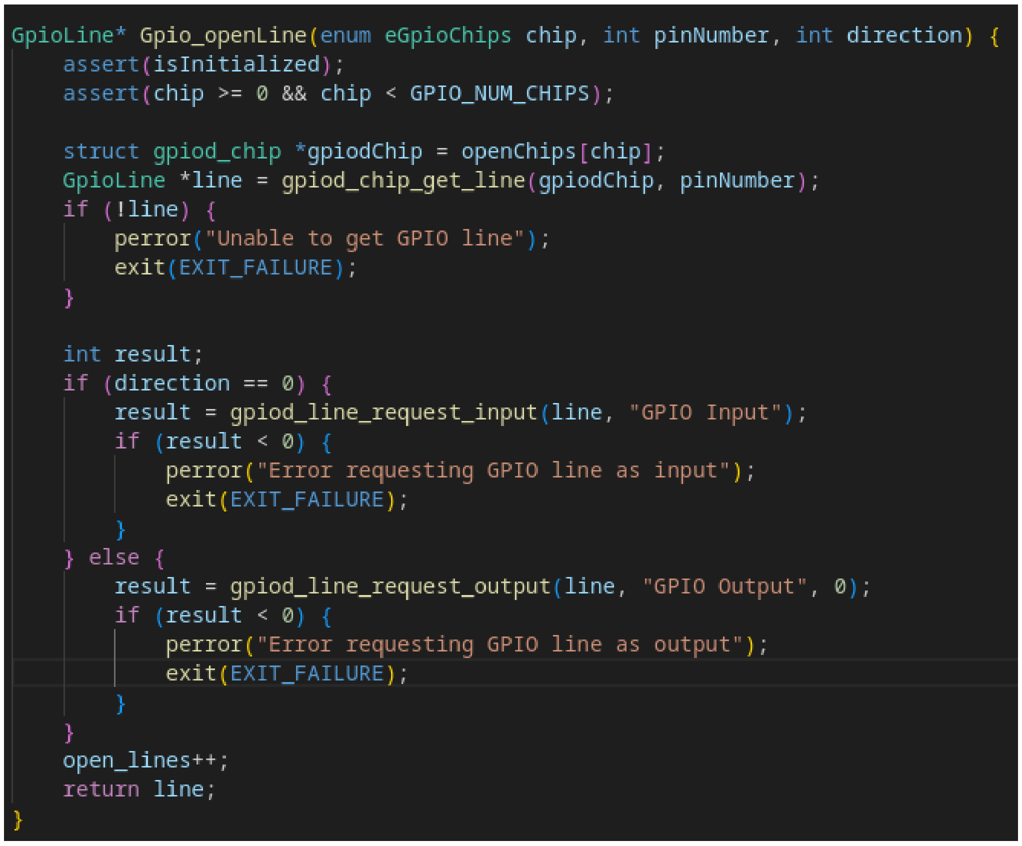
To read input from the keypad, you need to perform scanning.
This is where for every row pin, set it to high (1) then read each of the column pins. If the reading is a 1, that row, column pair button has been pressed.
This video explains the process clearly:
Raspberry Pi Pico Tutorial - 4x4 Matrix Keypad - MicroPython

Consider using gpioget and gpioset to test GPIO pins before coding the scanning.

Additional hints and troubleshooting can be found below.

Hints (C Code)

You will need to edit or add another function to gpio.c to handle opening certain lines as input and output.

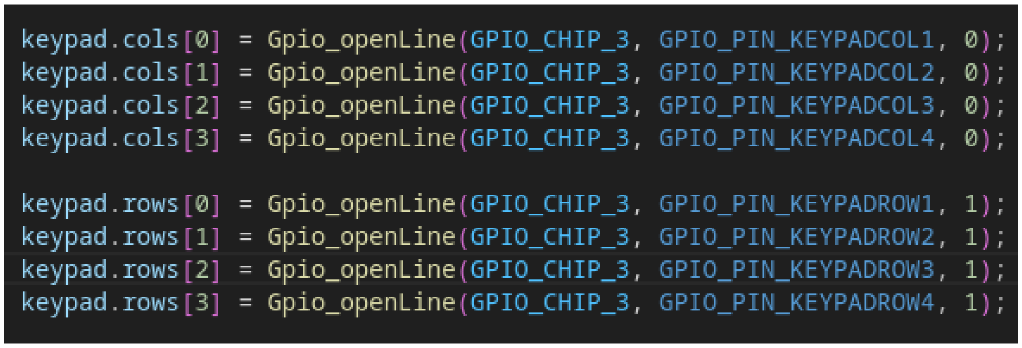


```c
GpioLine* Gpio_openLine(enum eGpioChips chip, int pinNumber, int direction) {
    assert(isInitialized);
    assert(chip >= 0 && chip < GPIO_NUM_CHIPS);

    struct gpiod_chip *gpiodChip = openChips[chip];
    GpioLine *line = gpiod_chip_get_line(gpiodChip, pinNumber);
    if (!line) {
        perror("Unable to get GPIO line");
        exit(EXIT_FAILURE);
    }

    int result;
    if (direction == 0) {
        result = gpiod_line_request_input(line, "GPIO Input");
        if (result < 0) {
            perror("Error requesting GPIO line as input");
            exit(EXIT_FAILURE);
        }
    } else {
        result = gpiod_line_request_output(line, "GPIO Output", 0);
        if (result < 0) {
            perror("Error requesting GPIO line as output");
            exit(EXIT_FAILURE);
        }
    }
    open_lines++;
    return line;
}
```

Initialize the column pins as input and row pins as output.

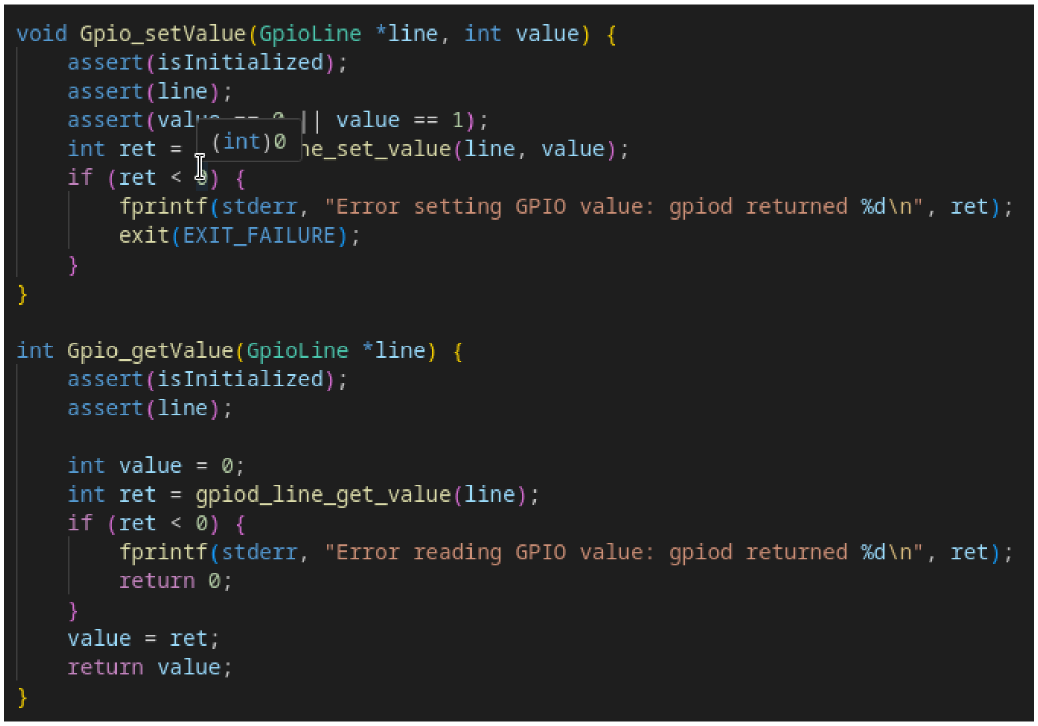


```c
keypad.cols[0] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADCOL1, 0);
keypad.cols[1] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADCOL2, 0);
keypad.cols[2] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADCOL3, 0);
keypad.cols[3] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADCOL4, 0);

keypad.rows[0] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADROW1, 1);
keypad.rows[1] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADROW2, 1);
keypad.rows[2] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADROW3, 1);
keypad.rows[3] = Gpio_openLine(GPIO_CHIP_3, GPIO_PIN_KEYPADROW4, 1);
```

Continued below…

Additional functions will need to be created to set certain values to pins when scanning.

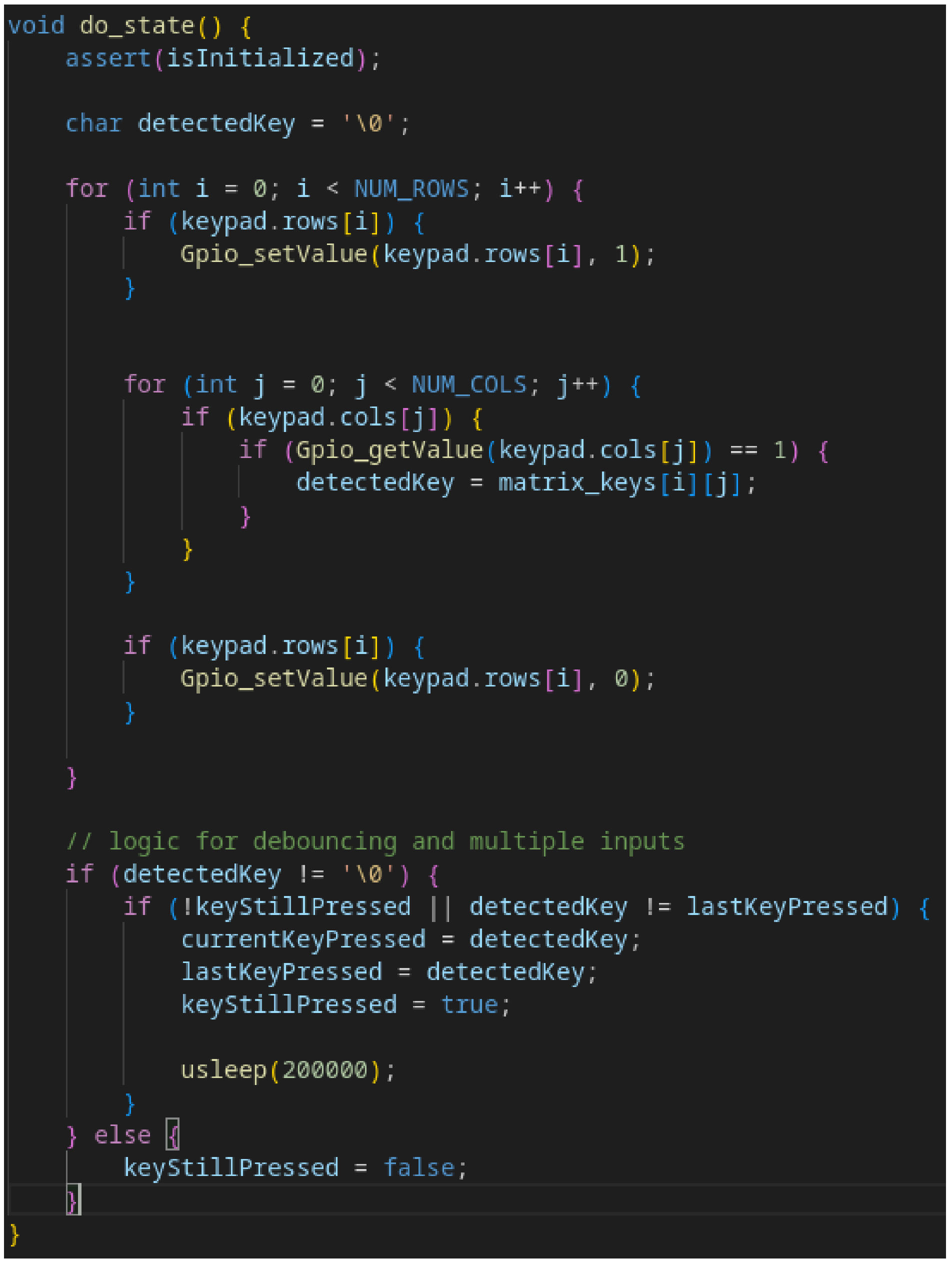


```c
void Gpio_setValue(GpioLine *line, int value) {
    assert(isInitialized);
    assert(line);
    assert(value == 0 || value == 1);
    int ret = (int)0 ne_set_value(line, value);
    if (ret < 0) {
        fprintf(stderr, "Error setting GPIO value: gpiod returned %d\n", ret);
        exit(EXIT_FAILURE);
    }
}

int Gpio_getValue(GpioLine *line) {
    assert(isInitialized);
    assert(line);

    int value = 0;
    int ret = gpiod_line_get_value(line);
    if (ret < 0) {
        fprintf(stderr, "Error reading GPIO value: gpiod returned %d\n", ret);
        return 0;
    }
    value = ret;
    return value;
}
```

Scanning logic can be found below.



```
void do_state() {
    assert(isInitialized);

    char detectedKey = '\0';

    for (int i = 0; i < NUM_ROWS; i++) {
        if (keypad.rows[i]) {
            Gpio_setValue(keypad.rows[i], 1);
        }


        for (int j = 0; j < NUM_COLS; j++) {
            if (keypad.cols[j]) {
                if (Gpio_getValue(keypad.cols[j]) == 1) {
                    detectedKey = matrix_keys[i][j];
                }
            }
        }

        if (keypad.rows[i]) {
            Gpio_setValue(keypad.rows[i], 0);
        }

    }

    // logic for debouncing and multiple inputs
    if (detectedKey != '\0') {
        if (!keyStillPressed || detectedKey != lastKeyPressed) {
            currentKeyPressed = detectedKey;
            lastKeyPressed = detectedKey;
            keyStillPressed = true;

            usleep(200000);
        }
    } else {
        keyStillPressed = false;
    }
}
```

Troubleshooting

1. Check all circuitry for correctness. Refer to the images in the document.
2. Make sure that when testing for input, the row pin(s) is set to high, then check the column pin.