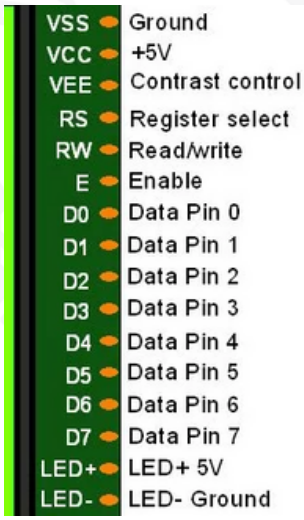
Digital bubble level

Basic prior knowledge on binary/hexadecimal is recommended

What is a LCD?

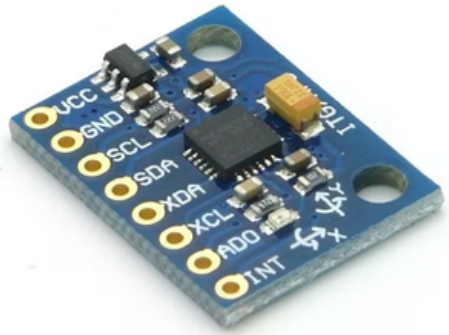
16x2 LCD and connections

- **LCD** stands for **L**iquid **C**rystal **D**isplay
- The one you'll be using has 2 rows of 16 characters.
- The method used to control what it displays is with binary comunication.
- This works by making a set of pins high or low representing a 0 or 1 in a sequence.
- The **LCD**s proccessor then uses these instructions to display something.



What is a IMU?

MPU 6050



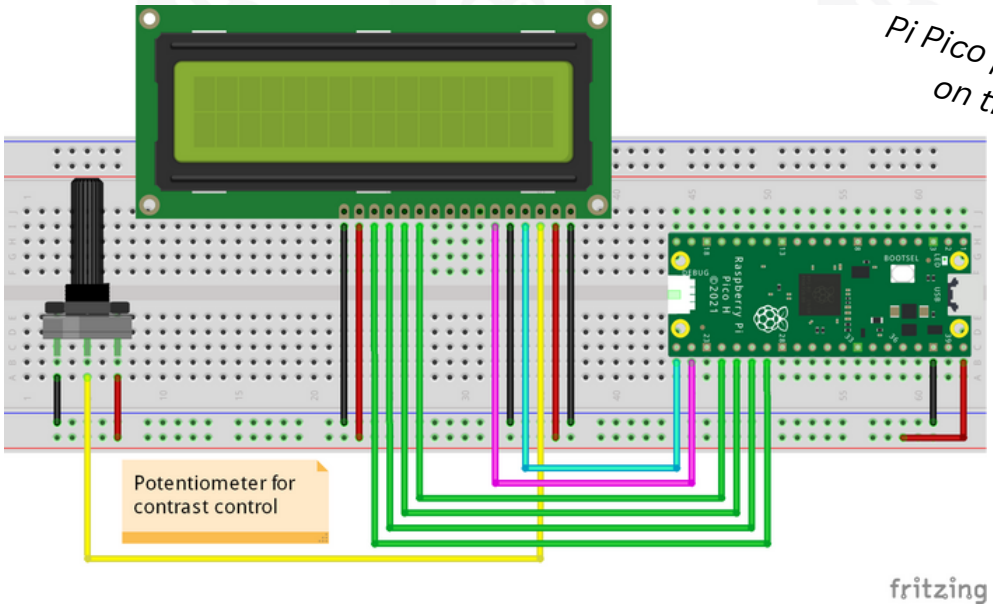
- An **IMU** is a **I**nertial **M**easurement **U**nit, also called an **MPU** (**M**otion **P**roccessing **U**nit)
- Its a device that can accurately measure 3-axis acceleration and rotation.
- It uses a protocol called **I2C** for communication, this works using 2 pins **SDA** (for data) and **SCL** (for clock) to transfer information.
- The Pi Pico has a built-in library for this protocol, and we'll use it to get motion data.

Part 1: Make the LCD say “Hello World!”

Please note the code for the LCD will be quite complex since we arent using any libraries, however the comments try to explain what is going on as best as possible.

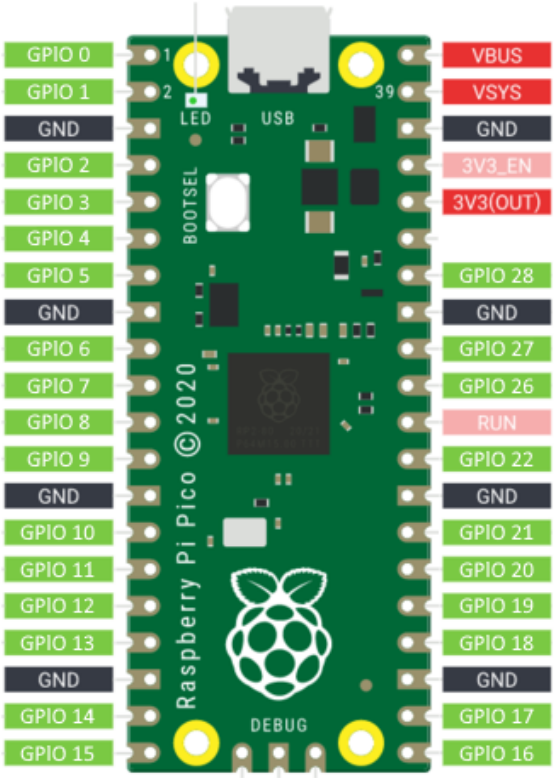
```
from machine import Pin
import utime  # were using utime instead of time because it allows for finer delay intervals

rs = Pin(16, Pin.OUT)  # LCD Reset pin
e = Pin(17, Pin.OUT)  # LCD enable pin

# We only need to use data pins 4 through 7 for basic functionality
d4 = Pin(18, Pin.OUT)  # LCD data pin 4
d5 = Pin(19, Pin.OUT)  # LCD data pin 5
d6 = Pin(20, Pin.OUT)  # LCD data pin 6
d7 = Pin(21, Pin.OUT)  # LCD data pin 7

dp = [d4, d5, d6, d7]  # store the data pins in an array for easier use

def pulseE():  # pulse a high signal on the enable pin, this tells the MPU to
    e.value(1)
    utime.sleep_us(40)
    e.value(0)
    utime.sleep_us(40)


def sendLCDByte(BinNum): # A function to send a byte of information to the LCD
    for a, b in zip(dp, [0x10, 0x20, 0x40, 0x80]): a.value(BinNum & b)
    pulseE()
    for a, b in zip(dp, [0x01, 0x02, 0x04, 0x08]): a.value(BinNum & b)
    pulseE()


def setUpLCD():  # initialize / reset the LCD's state
    utime.sleep_ms(100)  # wait a bit to allow the LCD to work when refreshing quickly
    rs.value(0)

    sendLCDByte(0b00110011)  # sending some configuration information
    sendLCDByte(0b00110010)

    sendLCDByte(0b00101000)
    sendLCDByte(0b00001100)

    sendLCDByte(0b00000110)
    sendLCDByte(0b00000001)

    utime.sleep_ms(2)  # small delay to allow settings to be applied on the LCD
    rs.value(1)


setUpLCD()  # runs the setup function
for x in "Hello World!": # converts each character to a number and sends it as a byte
    sendLCDByte(ord(x))
```

If you want to understand the sendLCDByte function further, feel free to ask!



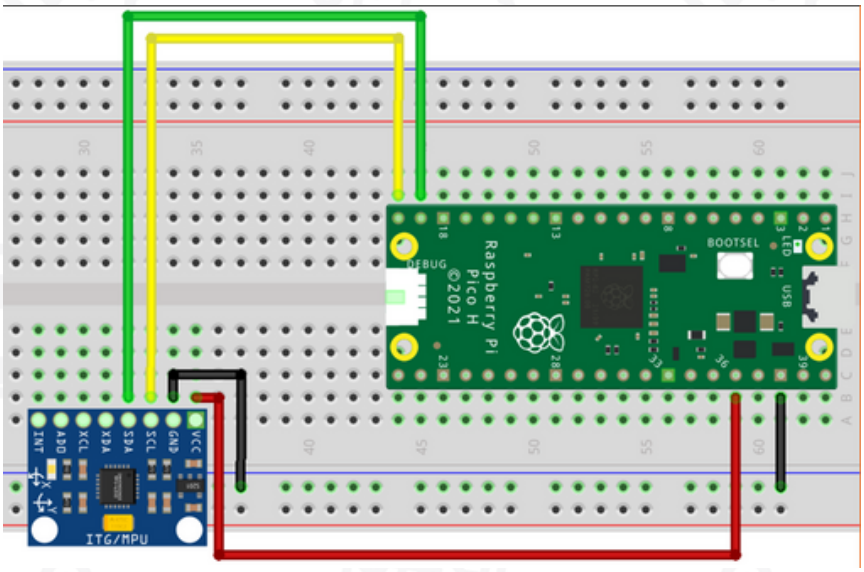
Pi Pico pinout diagram is on the next page

Part 2: Using the MPU6050



By now you should be seeing hello world on your LCD, well done!

Now we can move onto getting acceleration values from the IMU. The following circuit diagram doesnt include the LCD wiring to make it more readable, it doesnt mean you should remove what you have done so far.



*Note that the MPU6050 uses 3.3V **not** 5V*

To start add a few more imports and initialize the MPU:

```
from machine import Pin, I2C # Import the I2C Library
import utime, math  # Also import the math library

mpu = I2C(1, scl=Pin(15), sda=Pin(14), freq=100000)  # initialize I2C communication for the MPU
```

Now we can add send a wake-up message to the MPU and define a new function to convert 2 bytes into a 16-bit integer:

```
mpu.writeto_mem(104, 0x6B, bytes([0])) # Ensure the MPU isnt in sleep mode


def bytesToInt(msb, lsb):  # Function to convert two bytes into a 16-bit integer
    if not msb & 0x80:  # if most significant bit is 1, the integer is negative
        return msb << 8 | lsb
    return - (((msb ^ 255) << 8) | (lsb ^ 255) + 1)

```

You can use the line numbers to keep track of what code goes where

And at the end of your code, add a while loop to constantly read data:

```
while True:
    accBuffer = mpu.readfrom_mem(0x68, 0x3B, 6)  # Read the raw acceleration values from the MPUs memory

    accZ = bytesToInt(accBuffer[4], accBuffer[5]) # Convert the raw data to get an accel value for the z-axis

    print(f"accZ: {accZ}")  # print out the value to the console using a formatted string

    setUpLCD()
    for x in "Hello World!":  # print the value to the LCD
        sendLCDByte(ord(x))
```

Z-axis is vertical, so run your code, move your board and see the effect!

Task 3: Making a digital bubble level

So far you have read the z-axis value from the accelerometer, now you need to read all x, y and z axes to get a total magnitute (when board is level this will be gravity) and then calculate an angle between the board and the horizon to display on the LCD:

```
55
56 while True:
57     accBuffer = mpu.readfrom_mem(0x68, 0x3B, 6)  # Read the raw acceleration values from the MPUs memory
58
59     accX = bytesToInt(accBuffer[0], accBuffer[1]) # Convert the raw data to get an accel value for the x-axis
60     accY = bytesToInt(accBuffer[2], accBuffer[3]) # Convert the raw data to get an accel value for the y-axis
61     accZ = bytesToInt(accBuffer[4], accBuffer[5]) # Convert the raw data to get an accel value for the z-axis
62
63     totalMagnitude = math.sqrt((accX * accX) + (accY * accY) + (accZ * accZ));  # Pythagoras' Theorem
64     angle = round(math.acos(accZ/totalMagnitude) * 57.296) # Using trigonometry calculate the angle from hori
65
66     print(f"angle: {angle}")  # print out the value to the console using a formatted string
67
68     setUpLCD()
69     for x in "Angle: " + str(angle):  # print the value to the LCD
70         sendLCDByte(ord(x))
71
```

Run the code and Ta Da, you just made a digital bubble level!

Review and extra challenges:

Congrats! Today you have delved into the depths of digital communication and got a taste of what its like to interface with more complex components.

Challenge: Spruce up the LCD text by making the value centered.

Thinking point: We've seen what accelerometer (translational motion) data we get, but what values do you think we would get if we read the gyroscope (rotational motion)?

Pi Pico Code Docs

Worksheet produced by Lukas Hastings for use by HackSussex
https://github.com/supersand21/Robotics-Workshop