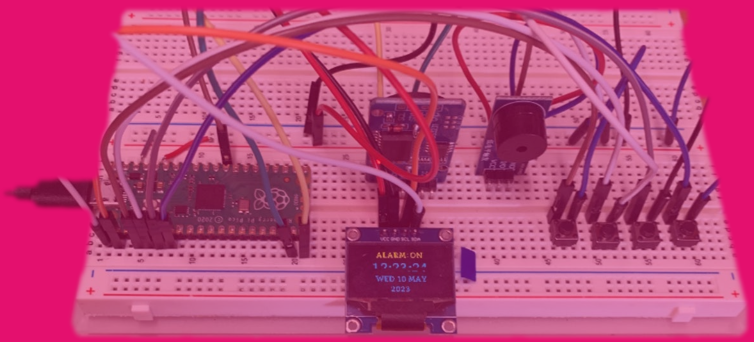
Let's build an Alarm Clock!

First, let's look at what functions our Alarm Clock is going to have

- Time and Date Display
- Alarm ON/OFF Function
- Set Time and Date Function
- Set Alarm Function
- An Alarm Sound that Should be very Familiar



What's in an Alarm Clock?

Most of us have used an alarm to get us out of bed in the morning, but have you thought what a digital alarm clock contains?

For an alarm clock to work, it requires a way for the clock to keep the correct time, a way for the time to be displayed, a way to interact with the clock, either via buttons or a touch screen, and some form of alarm.

Does the alarm need to make a sound? If not what other options could be used instead of sound.

University of Sunderland

Let's build an Alarm Clock!

What are we going to need to build our Alarm Clock?

KIT LIST

1 * D3231 RTC

1 * Piezo Buzzer

1 * Pi Pico

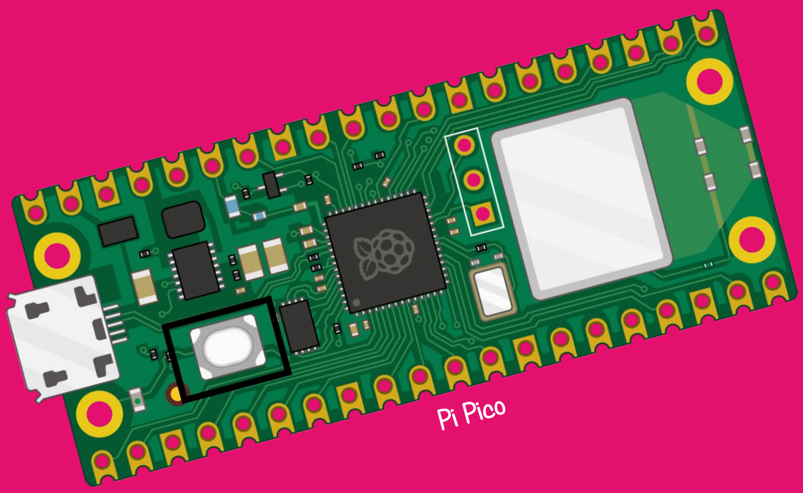
4 * Button switch

1 * oled 1306 Display

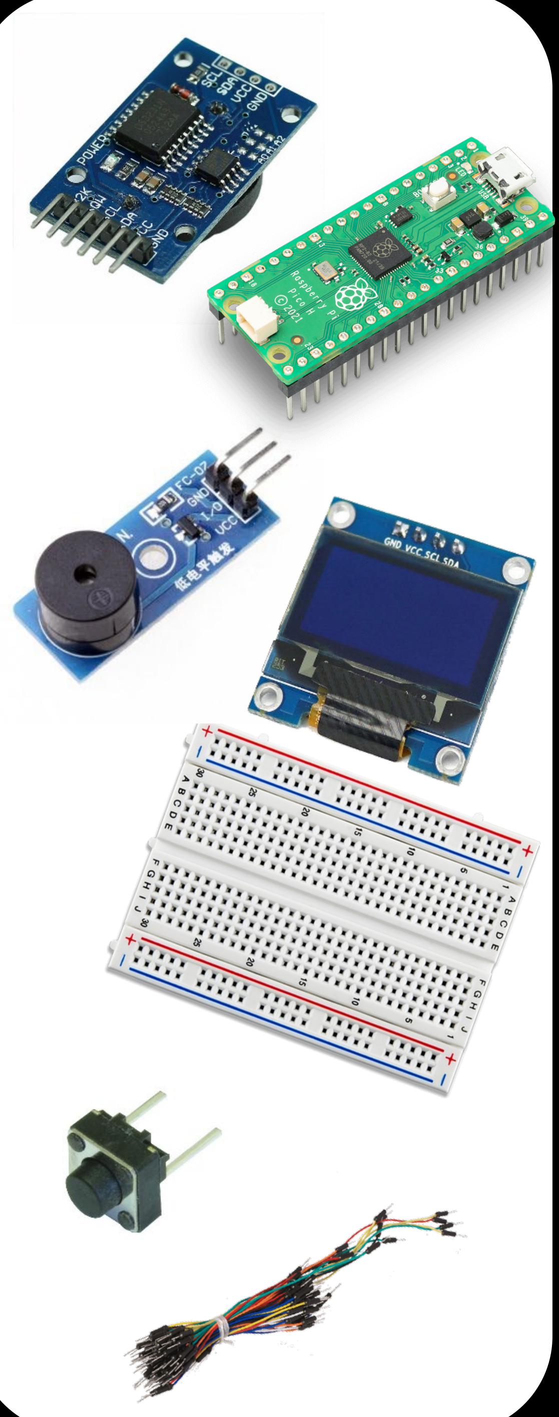
1 * Breadboard

11 * Jumper wires

1 * USB Cable



Kit



University of Sunderland

Let's build an Alarm Clock!

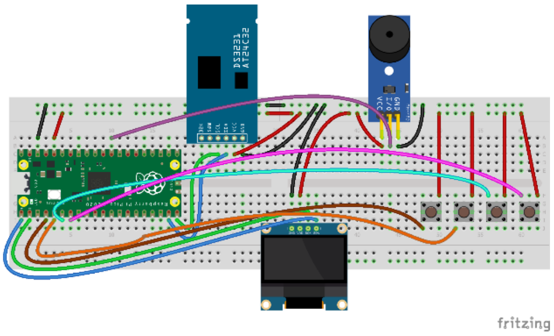
From design to build what will our Alarm Clock end up looking like?

KIT LIST

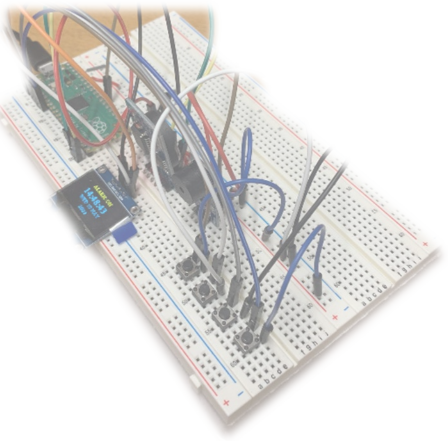
- 1 * D3231 RTC
- 1 * Pi Pico
- 1 * Piezo Buzzer
- 4 * Button switch
- 1 * oled 1306 Display
- 1 * Breadboard
- 11 * Jumper wires
- 1 * USB Cable

Faculty of Technology

Design

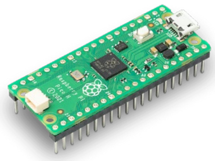


Build



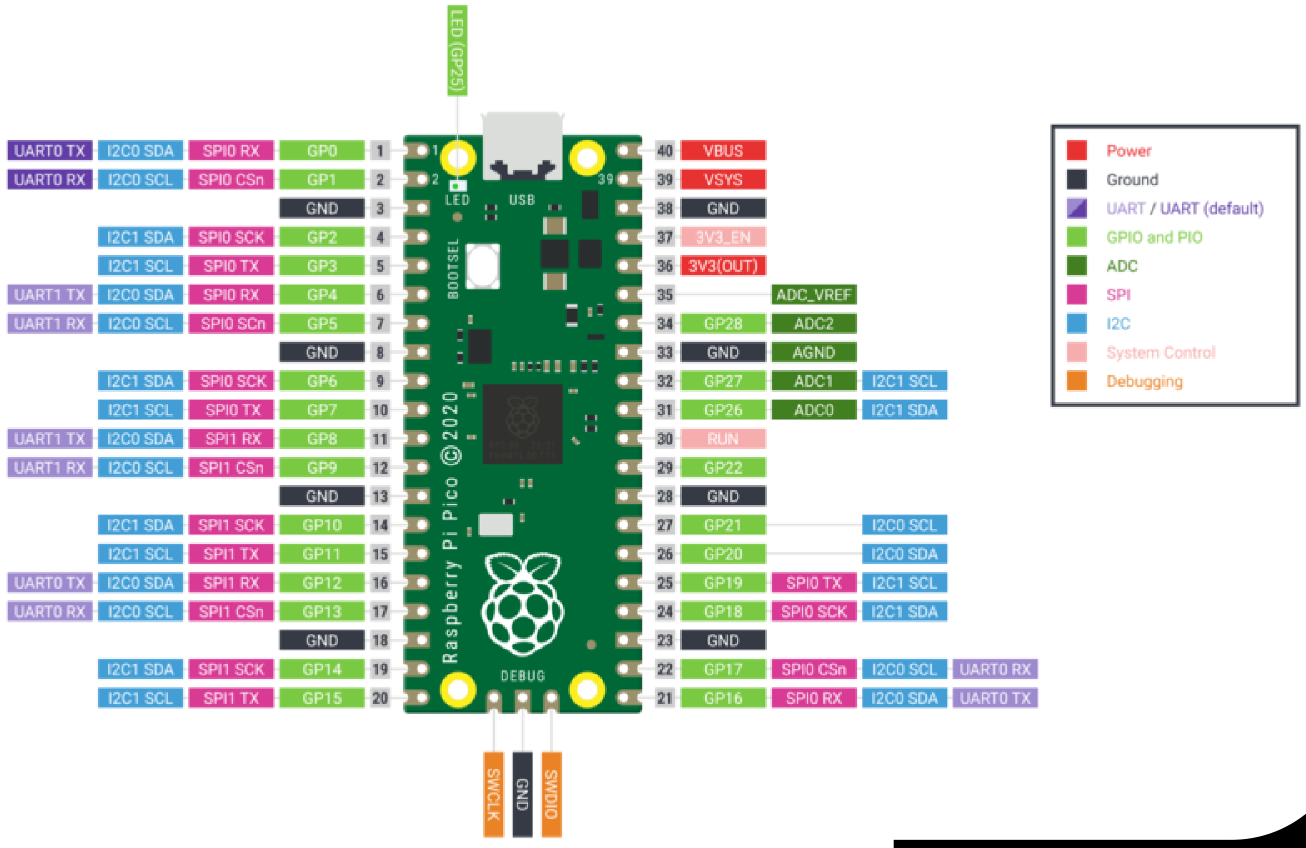
University of Sunderland

Let's build an Alarm Clock!



We're using a Raspberry Pi Pico as our microprocessor. So, what exactly is it?

Raspberry Pi Pico is Raspberry Pi's first microcontroller board, designed especially for physical computing. Microcontrollers are a different type of device than Single Board Computers (like the Raspberry Pi 4 and previous generations of Pi). They don't run an operating system and they are typically programmed to do just one task - though that task can be pretty intricate and exciting! They're perfect for experimenting with hardware and using as the brains of custom devices, machines, and inventions.

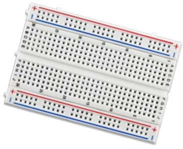


UART0 TX	I2C0 SDA	SPI0 RX	GP0	1
UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2
			GND	3
	I2C1 SDA	SPI0 SCK	GP2	4
	I2C1 SCL	SPI0 TX	GP3	5
UART1 TX	I2C0 SDA	SPI0 RX	GP4	6
UART1 RX	I2C0 SCL	SPI0 SCn	GP5	7
			GND	8
	I2C1 SDA	SPI0 SCK	GP6	9
	I2C1 SCL	SPI0 TX	GP7	10
UART1 TX	I2C0 SDA	SPI1 RX	GP8	11
UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12
			GND	13
	I2C1 SDA	SPI1 SCK	GP10	14
	I2C1 SCL	SPI1 TX	GP11	15
UART0 TX	I2C0 SDA	SPI1 RX	GP12	16
UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17
			GND	18
	I2C1 SDA	SPI1 SCK	GP14	19
	I2C1 SCL	SPI1 TX	GP15	20

40	VBUS			
39	VSYS			
38	GND			
37	3V3_EN			
36	3V3(OUT)			
35		ADC_VREF		
34	GP28	ADC2		
33	GND	AGND		
32	GP27	ADC1	I2C1 SCL	
31	GP26	ADC0	I2C1 SDA	
30	RUN			
29	GP22			
28	GND			
27	GP21		I2C0 SCL	
26	GP20		I2C0 SDA	
25	GP19	SPI0 TX	I2C1 SCL	
24	GP18	SPI0 SCK	I2C1 SDA	
23	GND			
22	GP17	SPI0 CSn	I2C0 SCL	UART0 RX
21	GP16	SPI0 RX	I2C0 SDA	UART0 TX

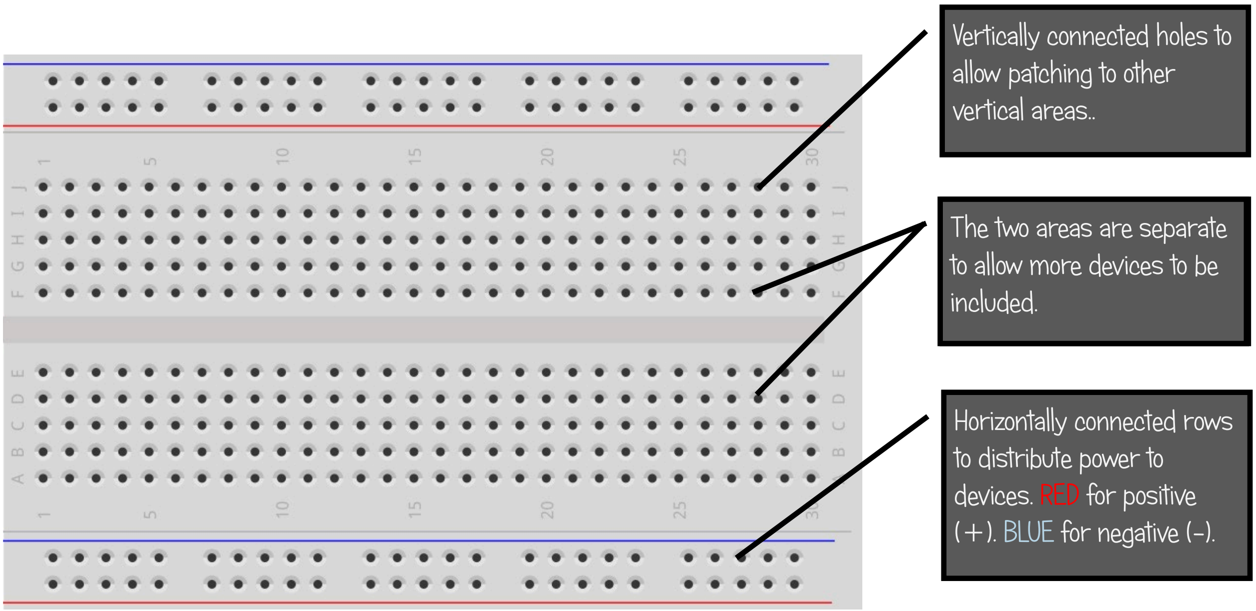
University of Sunderland

Let's build a Alarm Clock!



We're building our Alarm Clock as a prototype using a breadboard. So, what exactly is a breadboard?

A breadboard is used as a platform to prototype your circuits on. They provide a reusable environment that allows different ideas to be tested.
Breadboards come in different sizes, but they all have a pattern of holes for jumper wires. The main area has holes connected vertically and the edges (power rails) have the holes connected horizontally.

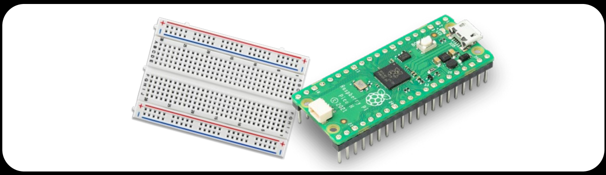


Vertically connected holes to allow patching to other vertical areas..

The two areas are separate to allow more devices to be included.

Horizontally connected rows to distribute power to devices. RED for positive (+). BLUE for negative (-).

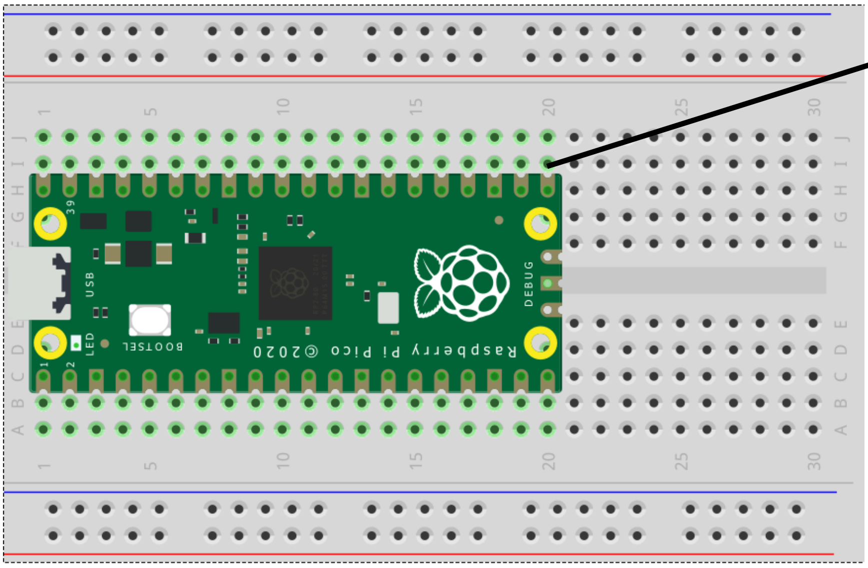
Let's build an Alarm Clock!



It's time to put the microprocessor onto the breadboard which is where we are going to build our Alarm Clock. So how do we line them up and fix them together?

The breadboard has 30/60 columns in two blocks in the centre. The Pico must be located at the very left-hand side with the microprocessor pins in the middle of the two blocks and let left most pins lined up to the very left-hand edge, as shown in the diagram below.

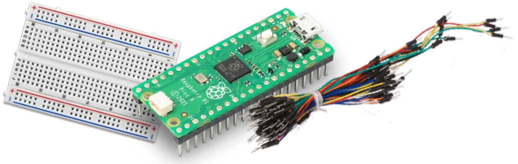
Be very careful pressing the Pico down, don't force it or it may break. Press gently at the edges so it sits flush to the breadboard with the Pico pins all the way in so you cannot see them.



Make sure there are 2 rows above and below the Pico.

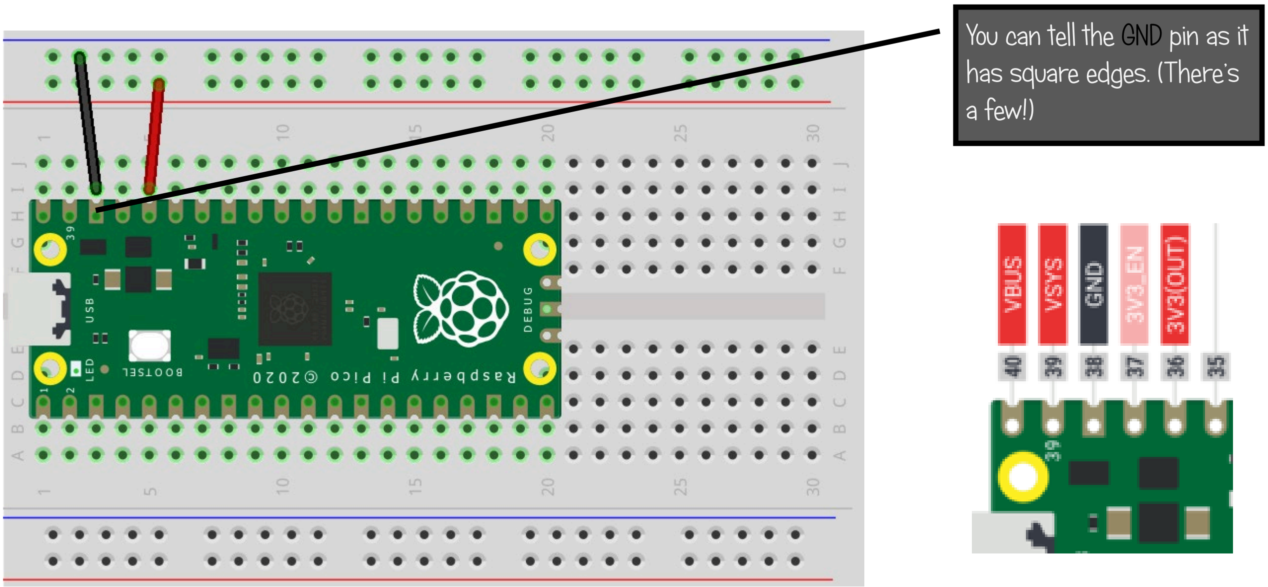
University of Sunderland

Let's build an Alarm Clock!



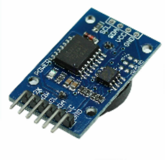
That's the Pico on the breadboard and we can start adding all the components and wires.
Let's start by adding power to the power rails (rows) at the top of the board.

The top row of the breadboard needs to have a 3.3v (+) and ground (GND) added to them.
We can do this by adding a wire from the 3.3v power out on Pico to the RED rail with a wire and a GND to the BLUE rail using a wire. See the diagram on page 4 for the Pins.
Pin 38 - BLUE rail. Pin 36—RED Rail



You can tell the GND pin as it has square edges. (There's a few!)

Let's build an Alarm Clock!



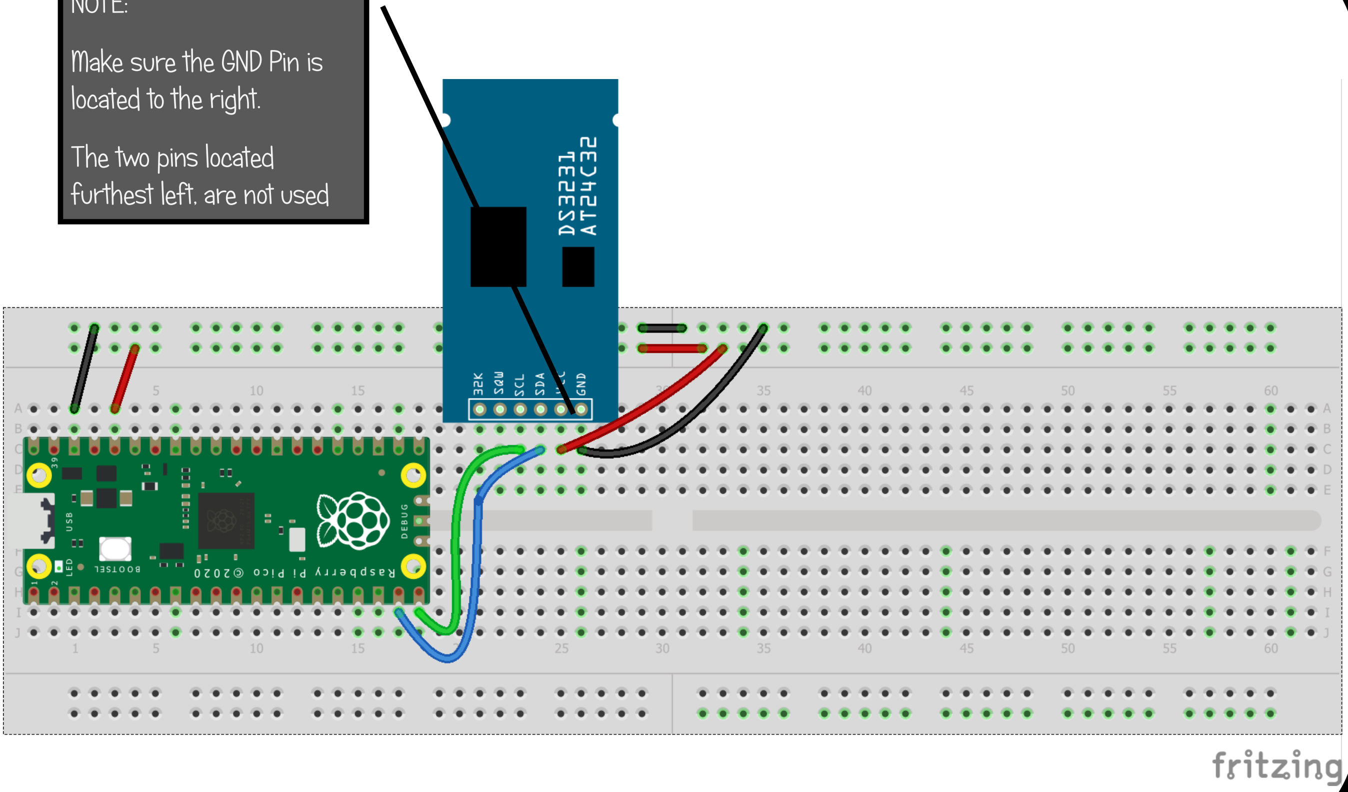
Now we have the basic prototype setup.
We have the Pi Pico on the breadboard
and the power distribution is ready.
Now we can site the rest of the components.

The first component we can add is the DS3231. Real Time Clock (RTC)
It has 6 connectors that need to be lined up on the breadboard. (but we only use 4 of them)
The four connectors are 3.3v power (VCC),
GND. the module Clock (SCL/SCK) and data signal (SDA) pins (in the middle) which send the readings to the Pico.

NOTE:

Make sure the GND Pin is located to the right.

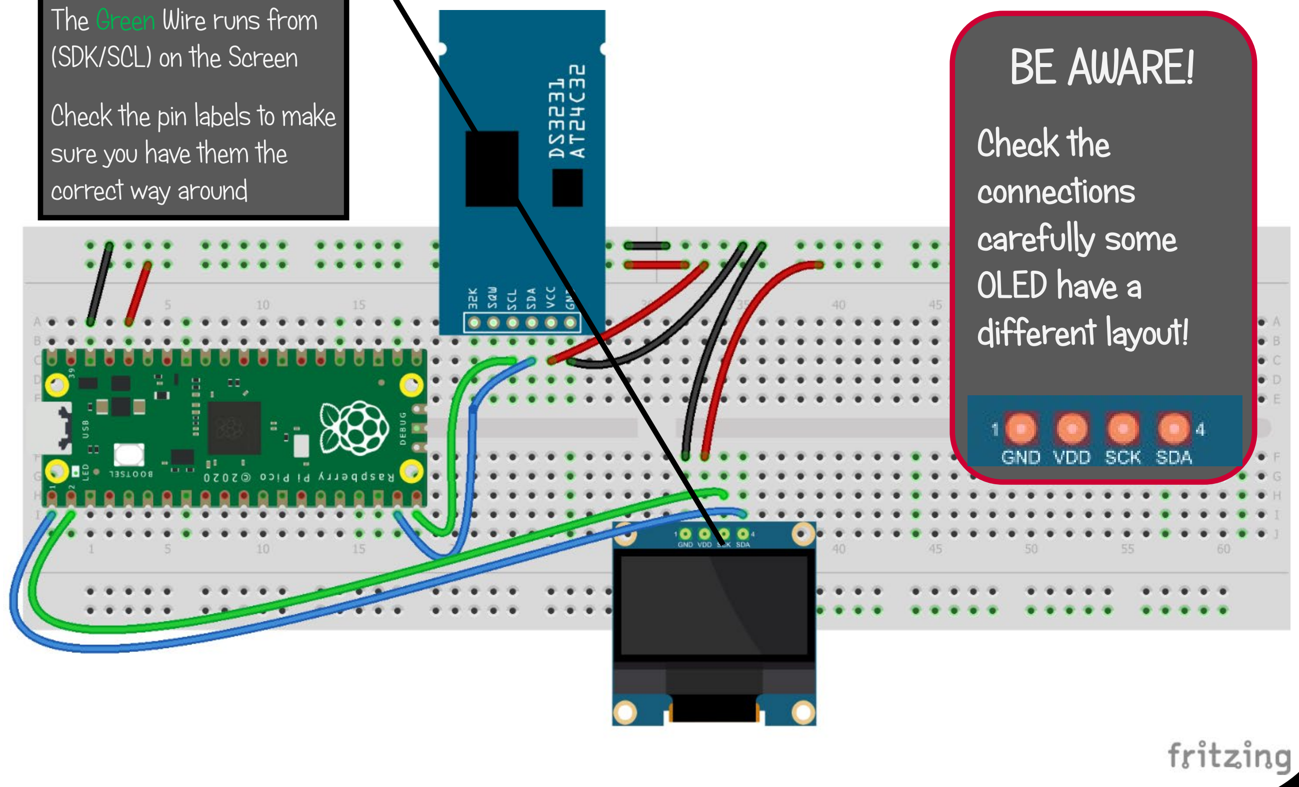
The two pins located furthest left. are not used



Let's build an Alarm Clock!

Now we have the basic prototype setup.
We have the Pi Pico on the breadboard
and the power distribution is ready.
Now we can site the rest of the components.

The next component we can add is the OLED screen.
It has 4 connectors that need to be lined up on the breadboard
The four connectors are 3.3v power (VCC), GND,
and the clock (SCL/SCK) and data signal (SDA) pins which send the readings to the Pico.



The Blue Wire Runs from SDA on the Screen

The Green Wire runs from (SDK/SCL) on the Screen

Check the pin labels to make sure you have them the correct way around

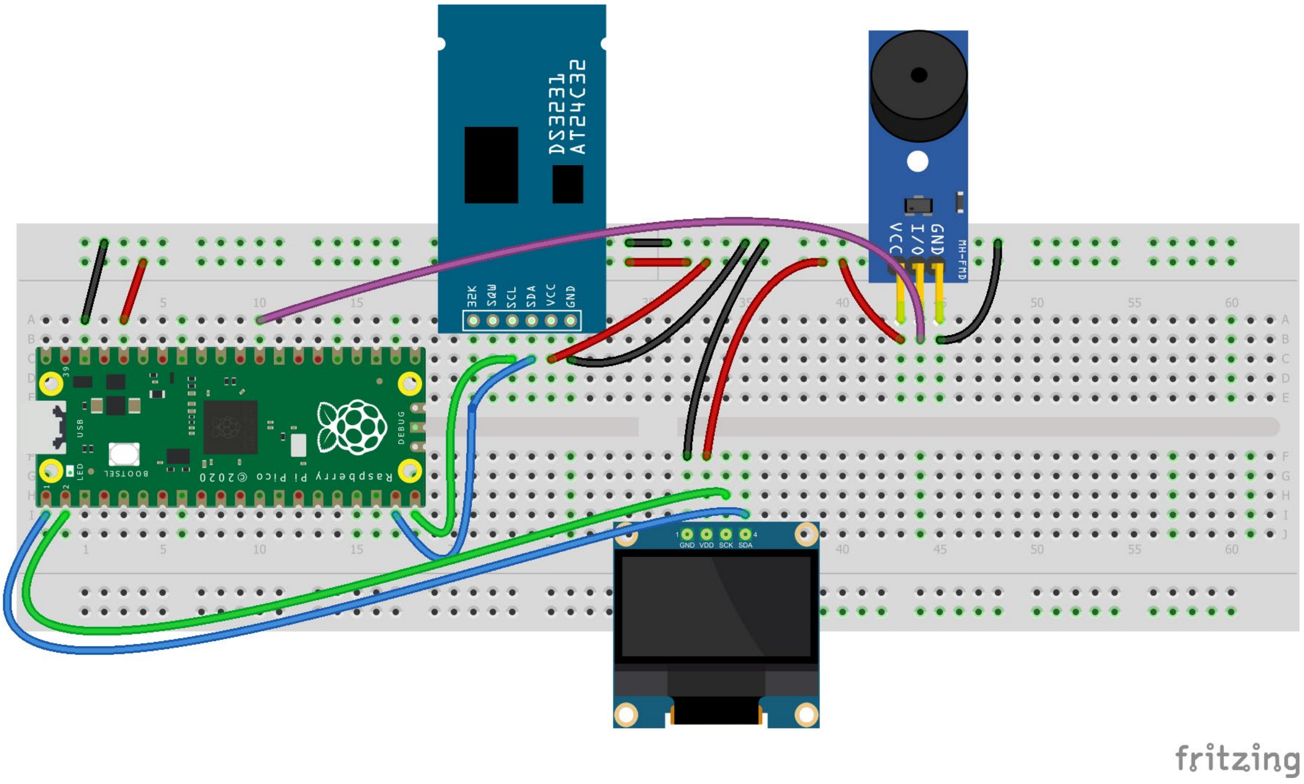
BE AWARE!

Check the connections carefully some OLED have a different layout!

Let's build an Alarm Clock!

Now we have the basic prototype setup.
We have the Pi Pico on the breadboard
and the power distribution is ready.
Now we can site the rest of the components.

The next component we can add is the Piezo Buzzer.
It has 3 connectors that need to be lined up on the breadboard
The three connectors are 3.3v power (VCC), GND,
and input/output (I/O)) which send the readings to the Pico.

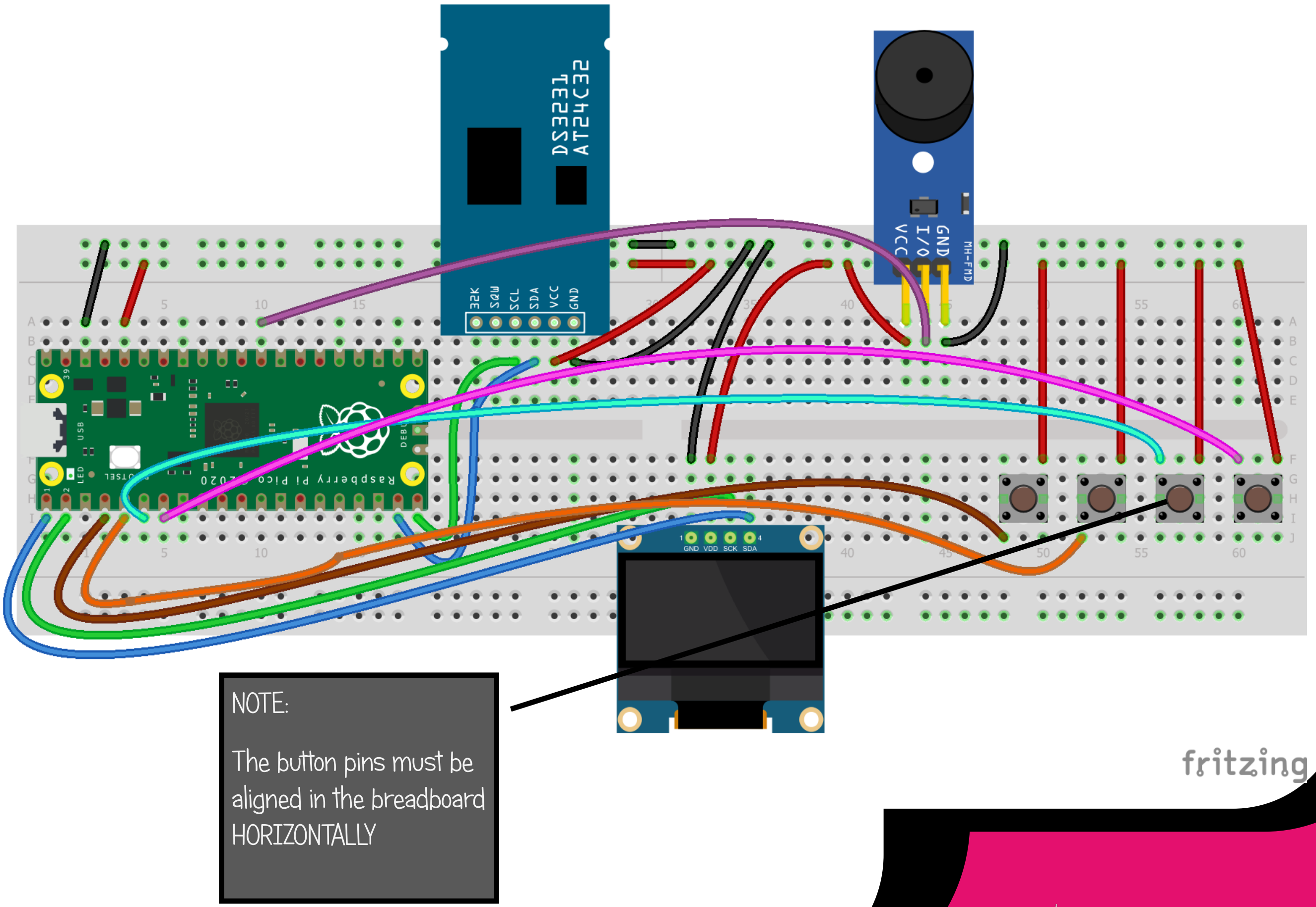


University of Sunderland

Let's build an Alarm Clock!

Now we have the basic prototype setup.
We have the Pi Pico on the breadboard
and the power distribution is ready.
Now we can site the rest of the components.

The next components we can add are the four Buttons.
They have 2 connectors that need to be lined up on the breadboard
These connectors need to be aligned horizontally on the breadboard.
Buttons will be numbered from 1 to 4 from left to right.

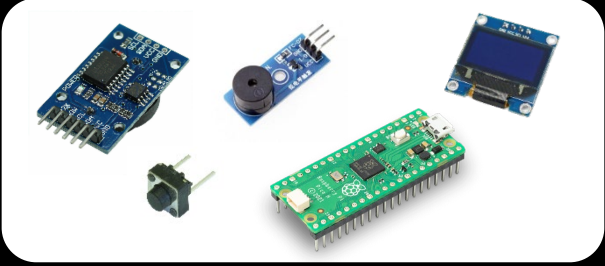


NOTE:

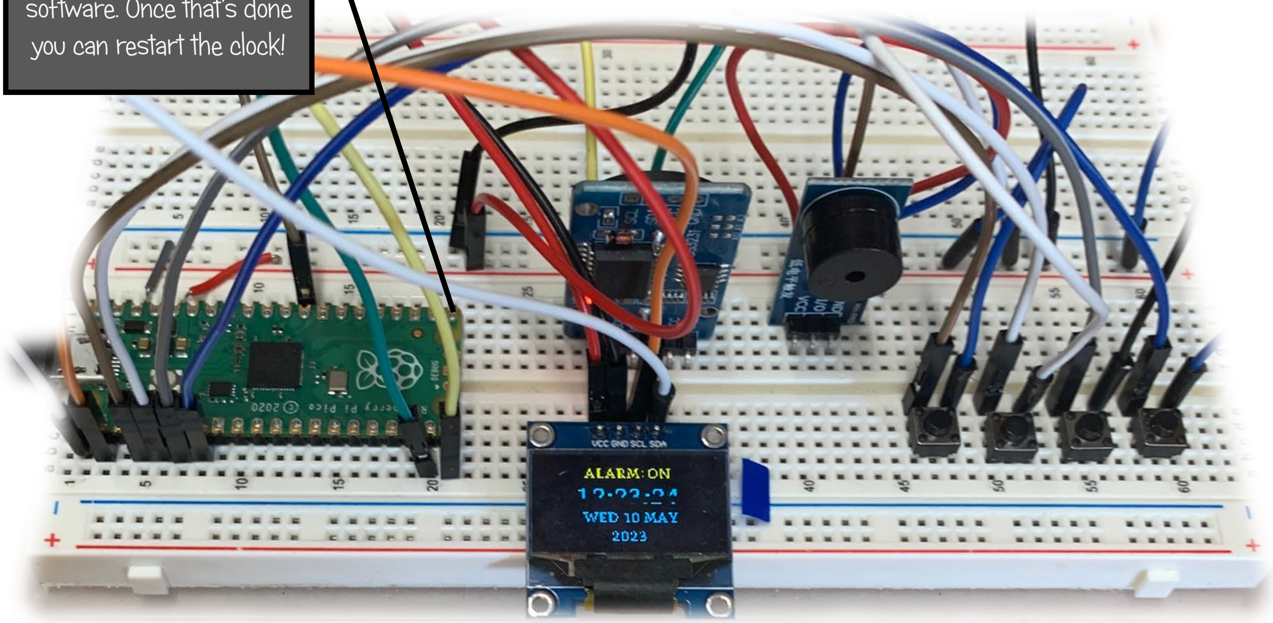
The button pins must be aligned in the breadboard HORIZONTALLY

University of Sunderland

Let's build an Alarm Clock!



Your Alarm Clock should now look something like this! All you need now is the code to load it onto the Pico with Thonny.



Now you need to use Thonny to install the software. Once that's done you can restart the clock!

Once the software is installed you can try placing it in different locations and see what happens!

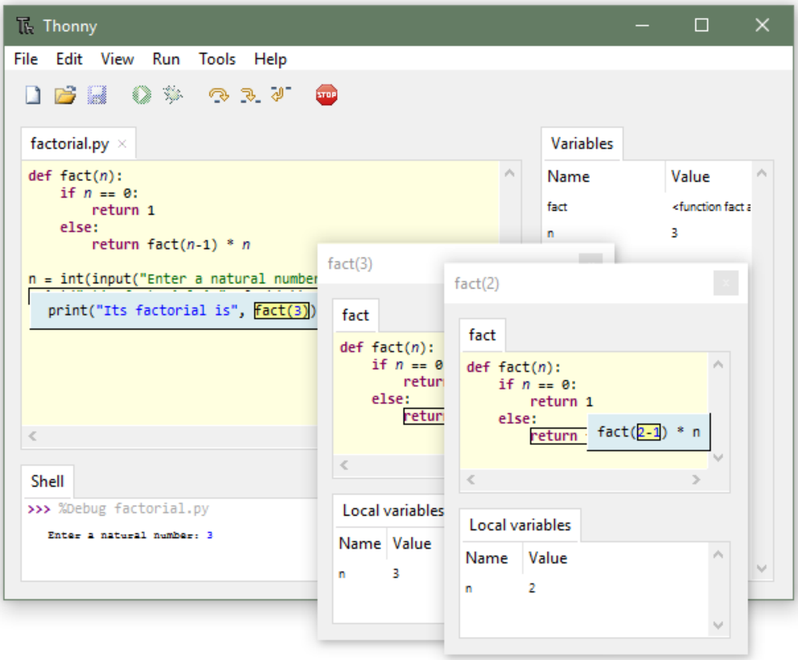
REMEMBER: You can use a USB power bank!

Let's build an Alarm Clock!

Follow the links below for the alarm clock code, how to download, and to get help on how to use Thonny.

Click HERE to get the code from GitHub!
Click HERE for the Thonny download.

Click HERE for how to install Micropython as a file.
Click HERE for how to install Micropython from Thonny.
Click HERE for help with using Thonny.



```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(n-1) * n

n = int(input("Enter a natural number
    print("Its factorial is", fact(3))
```

Name	Value
fact	<function fact a
n	3

```
>>> %Debug factorial.py
  Enter a natural number: 3
```

fact(3) — Local variables

Name	Value
n	3

fact(2)

```
def fact(n):
    if n == 0:
        return 1
    else:
        return fact(2-1) * n
```

Local variables

Name	Value
n	2

University of Sunderland

Let's build an Alarm Clock!

When Powering on the Alarm Clock, the time and date should be visible. However, these may be incorrect. Use the third button from the left (Button 3) to enter the Set Time Menu

Once In the Set Time Menu Use the Following Button Functions to Set the Time and Date

Buttons are numbered from Left to Right

Button 1: Increases the current value.

Button 2: Decreases the current value.

Button 3: Moves the cursor to the next value

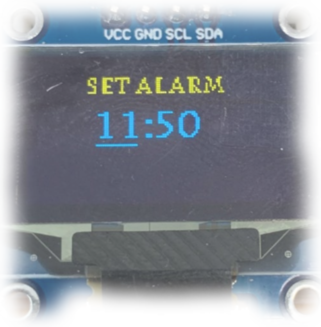
Button 4: Sets the current time and date values to memory

University of Sunderland

Let's build an Alarm Clock!

Once we have set the time and date, we can use Button 4 from the main display to enter the Set Alarm Time menu

†



Once In the Set Alarm Menu Use the Following Button Functions to Set the Time and Date

Buttons are numbered from Left to Right

Button 1: Increases the current value.

Button 2: Decreases the current value.

Button 3: Moves the cursor to the next value

Button 4: Sets the current displayed alarm time values to memory

University of Sunderland

Let's build an Alarm Clock!

Now That the Alarm is Set you can turn it ON/OFF from the main time display by pressing button 1, and can be cancelled by long pressing any button

Once In the Set Alarm Menu Use the Following Button Functions to Set the Time and Date

Buttons are numbered from Left to Right

Button 1: Turns Alarm ON/OFF.

Button 3: Enters the Set Time Menu

Button 4: Enters the Set Alarm Menu

NOTE: Once the Alarm is sounding you can turn it off by long pressing ANY button

Once the time matches the alarm set then a familiar theme tune should play.

May the force be with you

University of Sunderland

This page is intentionally left blank.

University of Sunderland