Maker Advent Calendar Day #10: Breaking Beams!

By The Pi Hut • Dec 9, 2022 • 18 comments

Welcome to day ten of your 12 Projects of Codemas Advent Calendar. Today we have another new sensor to play with - a **break beam sensor**!

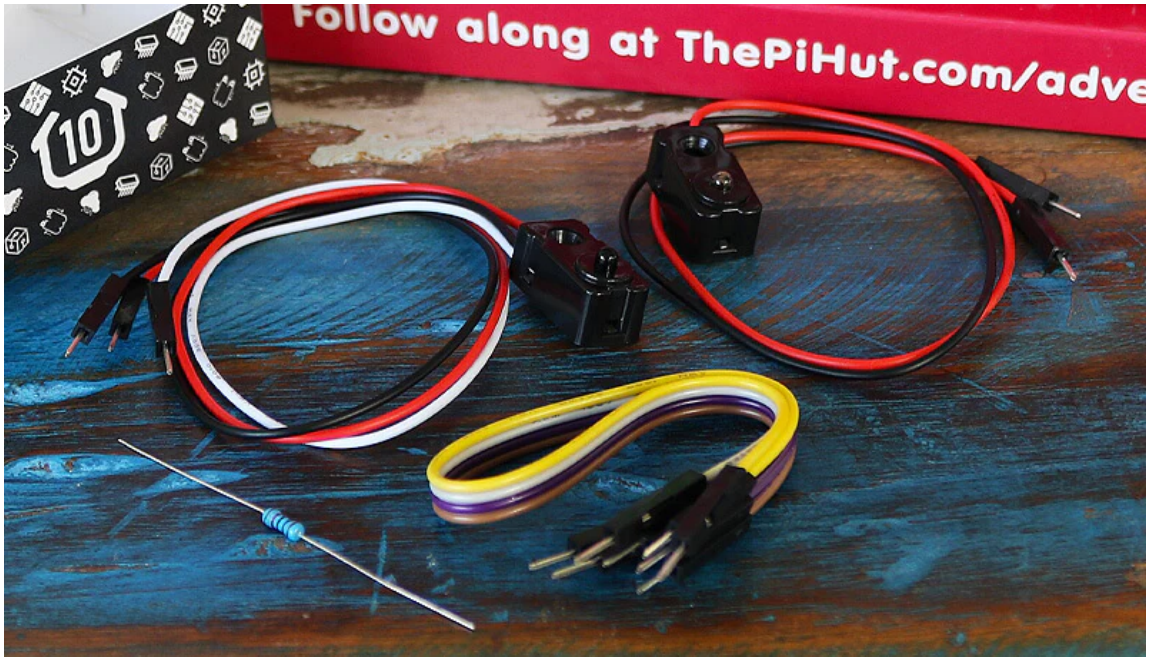
These sensors allow your project to detect something passing between them, so they're great as part of an alarm or security system, but can also be used to track the number of breaks for counting movement/rotations.

We're also going to make some fun games today so let's get straight into it!

Box #10 Contents

In this box you will find:

- **1x** Custom Break Beam Sensor (emitter and receiver) with male jumper wire ends
- **1x** 10k resistor
- **4x** Male to male jumper wires



Today's Project

Today we'll be getting our break beam sensor set up with a nice simple example, then jumping straight into a fun finger tap timer game!

The break beam sensor in your box comes with two parts - the **emitter** (2 wires) and the **receiver** (3 wires). The emitter sends an infra-red light out which is detected by the receiver, which keeps the white signal pin ***HIGH***. If something breaks this beam (*gets in the way*) the receiver does not detect a signal and then sends the pin ***LOW***.

We tell our code to watch for a ***LOW*** signal, which tells our program that something broke the beam, which we can then use to trigger an alarm, counter or another action. It's a *Reversy Percy* version of what we've used previously, as we usually look for ***HIGH*** signals.

These work well when placed up to 20-25cm apart, but they can be a little unreliable outside of that range. More advanced/industrial versions are used for larger areas such as doorways.

There are *so* many cool ways to use these:

- As a hand sensor for your desk to trigger your code
- With a train or Scalextric set, detecting when the train/car is passing to count laps
- A cookie jar movement alarm detecting when someone is sneaking a *cheeky biccy*
- With a robot, using a disk with a hole cut out to detect and count wheel revolutions
- ...and many more!

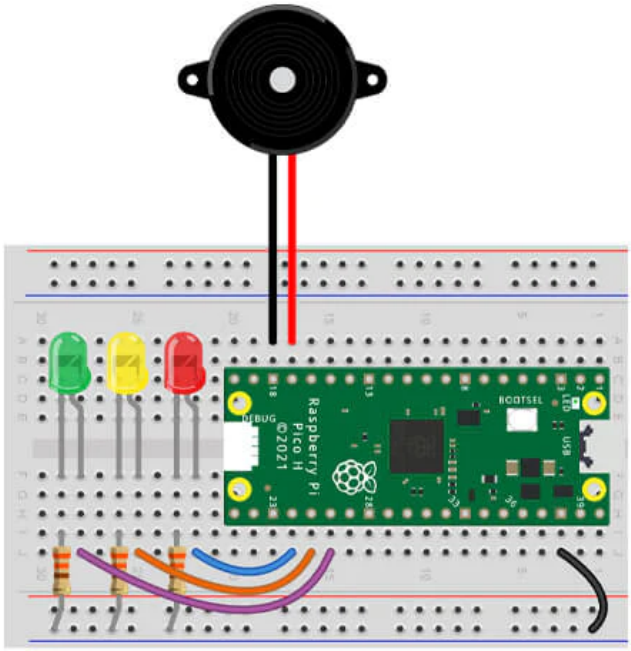
Construct the Circuit

As always, **unplug the USB cable so that your Pico isn't connected or powered up**.

Remove the tilt sensor parts from yesterday, but leave the LEDs and buzzer in place. We're not using the LEDs

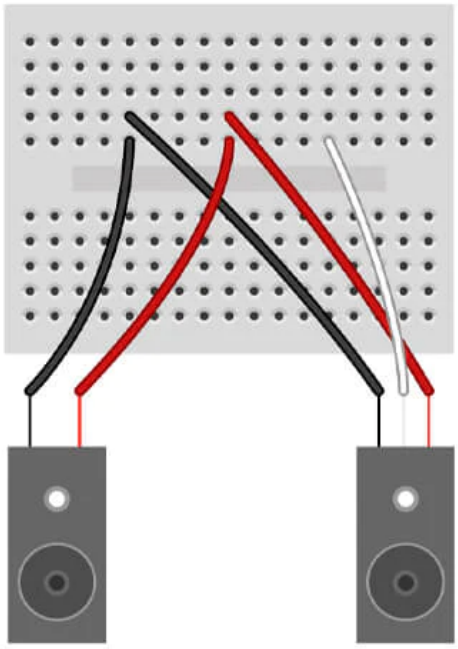
again after today's box!

Your starting circuit should look like this:

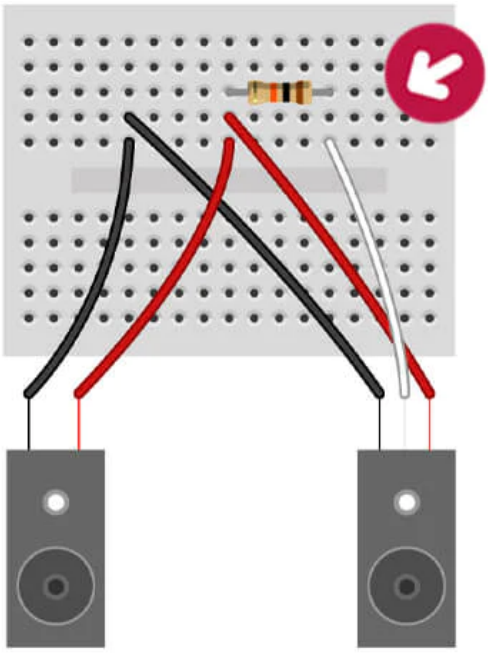


Now grab your mini breadboard and fit the ***red*** and ***black*** jumper wires into the same holes as we have below. The emitter and receiver will both use the same ***3.3V (red)*** and ***GND (black)*** pins, so we're fitting them to the same lanes here.

The ***white*** wire is the data/signal wire for the receiver only, so this sits to the side on its own:

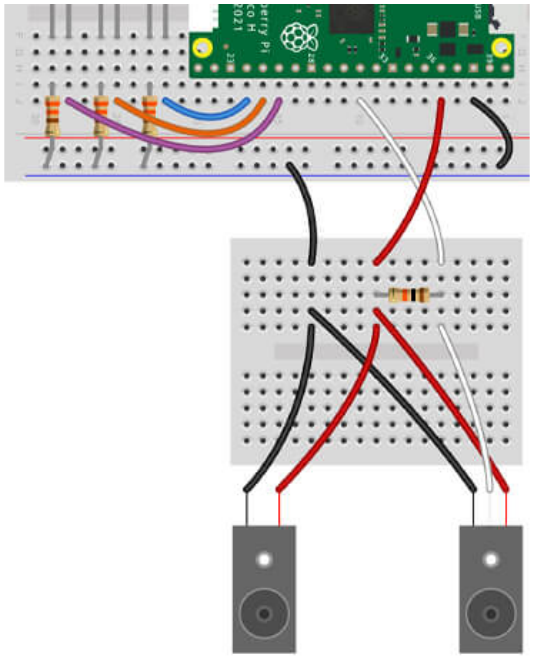


We need to add the included 10k resistor between the **3.3V** (***red***) and **signal** (***white***) pins, so fit this the same way as we have below:



Now we need to connect everything to our Pico/breadboard.

Connect the **black GND line to the blue GND** channel, the **red 3.3V line to our 3.3V pin** (***physical pin 36***) and the **white signal line to GPIO 26 (*physical pin 31*)**:



Activity 1: Basic Break Beam Program

We'll kick off today with a minimal break beam program to show how they work.

You'll need to secure the emitter and receiver on a flat surface with the little round heads facing each other, around ***15cm apart*** should work very well. You can use Blu-tack, sticky tape or simply some books or other heavy items resting on top to hold them in place - just make sure nothing is in the way of the beams.

You also need to keep an eye on the jumper wires as they like to escape from breadboards sometimes!

The Code

Nice and simple code here that you've played with before - we ***import*** what we need, set up the break beam pin number (with ***pull down***), then run a ***while loop*** with an ***if statement*** waiting for the pin to go ***LOW*** (the beam to be broken).

Copy this over to Thonny, run the code and then block the beam with your hand:

```
# Imports
from machine import Pin
import time

# Set up our beam pin
beam = Pin(26, Pin.IN, Pin.PULL_DOWN)

while True: # Run forever

    time.sleep(0.1) # Short delay

    if beam.value() == 0: # If the beam is broken

        print("Beam Broken!")
```

Activity 2: Break Beam High Score Game

Let's make a little game with these sensors that:

- Counts the number of times you can break the beam within 30 seconds
- Prints your score at the end
- Then closes the program

We need to ensure players can't cheat by simply holding their finger in place, so we'll use the same method as yesterday's counting code where we demand the pin status to change before the code will accept another count. We'll also add our buzzer to the code to add beeps for the game start/finish.

There are a few new elements in the example code that we need to introduce, so let's go over them first...

Time and Epoch

We've been using ***time.sleep*** a lot over the last ten days, however we're going to use another part of that module today.

In our example code we use ***time.time()***. This returns the number of seconds since ***epoch***. Epoch is a timestamp widely used in computing, and is the number of seconds that have elapsed since *1st January 1970!*

Now, this might seem like a *really* odd thing for us to use here, but as epoch time keeps ticking a second at a time, it's an easy to use, reliable and readily available timer. There are other ways to run timers, but this was a good excuse to talk about epoch!

We take a snapshot of epoch as our game start time, storing it in a ***variable***, then take readings every loop, using simple maths to compare the start time (in seconds) to the current time (in seconds), checking whether 30 seconds have passed since the start time.

We can't simply count seconds since the game start with our usual ***time.sleep*** as this doesn't account for other delays in our program (such as the short delay we add to the while statement to avoid the program running at a ridiculous speed).

If you're curious, try the very short code below to see what epoch currently looks like:

```
import time

print(time.time())
```

Sys

We also import ***sys*** in this example. The only reason we add this is to allow us to use ***sys.exit()*** to end the program after the game has finished.

It's another handy command you can add to your programs to ensure it ends after a condition is met. Like all things code/Python/MicroPython, there are many ways to achieve the same result - this is just one way we wanted to introduce.

The Code

Although the code example below is longer than some of the others we've shown you, there's nothing scary or new here (minus the notes above). We're importing, setting up pins and PWM, creating some variables then using a while loop with if statements - all things you're good at now!

You'll notice that our ***if statement*** always checks the timer first, because this is the most important thing. If the time is up, we don't want our player being able to continue. If the time isn't up, the ***elif*** statements are there to check for the state and whether the beam is broken.

Game On!

Copy the code below over to Thonny and give it a try, then see if you can beat some of our staff record times below (the only rule is that only a single finger can be used). It's a great little game to set up over the festive period to challenge family and friends, and you get to show off your new kit and skills too!

- Billie: **223**
- Marie: **221**
- Rich: **220**
- Chris C: **190**
- Jamie: **183**
- Colin: **172**
- Nigel: **165**

```
# Imports
```

```
from machine import Pin, PWM
import time, sys

# Set up the Break Beam pin
beam = Pin(26, Pin.IN, Pin.PULL_DOWN)

# Set up the Buzzer pin as PWM
buzzer = PWM(Pin(13))

# Set buzzer PWM frequency to 1000
buzzer.freq(1000)

# Start with the buzzer volume off (duty 0)
buzzer.duty_u16(0)

# Create game variables
starttime = 0
timecheck = 0
scorecounter = 0
state = 0

print("Game starts after the beep!")

#Long beep to signal game start
buzzer.duty_u16(10000)
time.sleep(2)
buzzer.duty_u16(0)

print("GO!")

# Store our start time (seconds)
starttime = time.time()

while True: # Run this block until code stopped

    time.sleep(0.0001) # Very short delay

    # Take the current epoch time and minus the start time
    # This gives us the number of seconds since we started the game
    timecheck = time.time() - starttime

    if timecheck >= 30: # If 30 or more seconds have passed

        print("GAME OVER!")

        # Beep to signal game end
        buzzer.duty_u16(10000)
        time.sleep(0.2)
        buzzer.duty_u16(0)

        # Print the player's score
        print("YOUR SCORE:",scorecounter)
```

```
        # Exit the program
        sys.exit()

    elif state == 0 and beam.value() == 0: # If state is 0 AND our pin is LOW

         scorecounter = scorecounter + 1 # Add +1 to our score counter

         state = 1 # Change state to 1

         print("SCORE =",scorecounter) # Print our new score counter
         print("Time remaining:", (30 - timecheck)) # take our timecheck variable awa

    elif state == 1 and beam.value() == 1: # If state is 1 AND our pin is HIGH

        state = 0 # Change the state to 0
```

Activity 3: Break Beam Target Score Game

The last activity created a game where we tried to hit the highest score possible in 30 seconds. This time we're creating a game with a target to hit within 30 seconds, using our LEDs to indicate how close the player is to beating the game.

It uses similar code to the example above, but we make it a little busier by adding in the LEDs with a new variable to set a target (***targetscore***), additional ***if statements*** to light the LEDs depending on score, and some tweaks to the ***print*** lines to show the target next to their current score.

Setting a target score

To allow us to easily set a different target score (which drives the game and print lines), we set the target as a ***variable*** and use that throughout the program. We even use this to drive the LEDs by turning them into percentages in our ***if statements***.

For example, one of the LED if statements looks like this:

```
if scorecounter < (targetscore / 100 * 33):
```

This says "***If the players score is less than 33% of the target score***".

All we're doing here is taking our target score, dividing it by 100 to give us 1%, then multiplying that by whatever percentage of score completion we want each LED to light at - and then compare that to the players score.

We use ***0 to 33%*** **for the red LED**, ***33% to 66% for the amber LED***, then any score ***over 66% lights the green LED***. This gives our player a visual indication of how close they are to beating the game.

Ifs in ifs?!

You'll notice the LED indicator ***if statements*** are sitting ***inside an elif statement***. This is fine and is known as a ***nested if statement***. Whilst it can make your code a little trickier to read and review, it's another tool you can use if you want the triggered if statement to then check something else.

In our example, we're happy that the player triggered the elif statement to increase their score, but we also want to check the score and update our LEDs as part of that.

The Code

Give it a spin by copying and running the code below, then why not try:

- Increasing the target score to make the game harder
- Changing the LED percentages/ranges
- Creating ***functions*** to turn the LEDs off or beep the buzzer, to save a few lines of code

```
# Imports
from machine import Pin, PWM
import time, sys

# Set up LED pins
red = Pin(18, Pin.OUT)
amber = Pin(19, Pin.OUT)
green = Pin(20, Pin.OUT)

# Set up the Break Beam pin
beam = Pin(26, Pin.IN, Pin.PULL_DOWN)

# Set up the Buzzer pin as PWM
buzzer = PWM(Pin(13))

# Set buzzer PWM frequency to 1000
buzzer.freq(1000)

# Start with the buzzer volume off (duty 0)
buzzer.duty_u16(0)

# Create game variables
starttime = 0
timecheck = 0
scorecounter = 0
state = 0
targetscore = 100

print("Game starts after the beep!")

#Long beep to signal game start
buzzer.duty_u16(10000)
time.sleep(2)
buzzer.duty_u16(0)

print("GO!")
print("--------------------------------")

# Store our start time (seconds)
starttime = time.time()
```

```
while True: # Run this block until code stopped

    time.sleep(0.0001) # Very short delay

    # Take the current time and minus the original start time
    # This gives us the number of seconds since we started the game
    timecheck = time.time() - starttime

    if timecheck >= 30: # If 30 or more seconds have passed

        # LEDs off
        red.value(0)
        amber.value(0)
        green.value(0)

        # Beep to signal game end
        buzzer.duty_u16(10000)
        time.sleep(0.2)
        buzzer.duty_u16(0)

        # Print the target and player's score
        print("--------------------------------")
        print("GAME OVER! YOU LOSE :(")
        print("The target was",targetscore,", you scored",scorecounter)
        print("--------------------------------")

        # Exit the program
        sys.exit()

    elif scorecounter >= targetscore: # If player's score has hit the target

        # LEDs off
        red.value(0)
        amber.value(0)
        green.value(0)

        # Beep to signal game end
        buzzer.duty_u16(10000)
        time.sleep(0.2)
        buzzer.duty_u16(0)

        # Print time taken to win
        print("--------------------------------")
        print("YOU WIN!")
        print("You took",timecheck,"seconds!")
        print("--------------------------------")

        # Exit the program
        sys.exit()

    elif state == 0 and beam.value() == 0: # If state is 0 AND our pin is LOW

         scorecounter = scorecounter + 1 # Add +1 to our score counter
```

```
            state = 1 # Change state to 1

            print("SCORE =",scorecounter,"/",targetscore) # Print the score and target
            print("Time remaining:", (30 - timecheck)) # take our timecheck variable awa

            if scorecounter < (targetscore / 100 * 33): # If our score is less than 33%

                red.value(1) # Red LED on
                amber.value(0)
                green.value(0)

            elif (targetscore/ 100 * 33) < scorecounter < (targetscore / 100 * 66): # If

                red.value(1) # Red LED on
                amber.value(1) # Amber LED on
                green.value(0)

            elif scorecounter > (targetscore / 100 * 66): # If our score is over 66% of

                red.value(1) # Red LED on
                amber.value(1) # Amber LED on
                green.value(1) # Green LED on

        elif state == 1 and beam.value() == 1: # If state is 1 AND our pin is HIGH

            state = 0 # Change the state to 0
```

Day #10 Complete!

We hope you enjoyed the games in today's break beam activities - why not get the whole family involved to show off your project and see who can score the highest for ultimate bragging rights?

You now have another sensor in your ever-growing box of bits to make projects with - perhaps you'll combine it with the PIR sensor from day #7 to make an awesome alarm with multiple sensors?

This is the last time we'll be using the LEDs in the calendar, so go ahead and remove them from your circuit (it's best to keep them in the same bag as the resistors so that you don't mix them up). We also won't be using the buzzer tomorrow, so you can remove that as well.

So what did we cover on day #10? Today you have:

- Built a circuit with break beam sensors
- Learnt how to code break beams sensors (*with their 'always HIGH' way of working*)
- Created a fun break beam game
- Learnt additional time module options, including epoch time
- Learnt how to use the sys module to end a program
- Created nested if statements
- Used simple maths to turn values into percentages with MicroPython

- Continued to re-use all off the things you've learnt over the last 10 days

Tomorrow **does not include a sensor!** We figured you might be a bit '*sensored out*' by now. This was the last sensor you will receive in the calendar, *aaaand now you're wondering what's in the next two boxes*...see you tomorrow!

Featured Products

Maker Advent Calendar - The 12 Projects of Codemas (inc. Raspberry Pi Pico H)

£40 incl. VAT

ADD TO CART

18 comments

The Pi Hut

December 1, 2023 at 8:48 am

@Matthew – Oh that's odd, maybe a faulty unit – sorry :(Please get in touch with our support guys at support.thepihut.com and they will help with that.

@Matthew – Oh that's odd, maybe a faulty unit – sorry :(Please get in touch with our support guys at support.thepihut.com and they will help with that.

Matthew

December 1, 2023 at 8:46 am

I think my IR emitter isn't working, I've tested it by looking at it while powered though my phone camera and I can't see anything light up. I'm also having the same issue where the Receiver only goes low when the sensor is completely obscured at touching distance.

I think my IR emitter isn't working, I've tested it by looking at it while powered though my phone camera and I can't see anything light up. I'm also having the same issue where the Receiver only goes low when the sensor is completely obscured at touching distance.

Jon Taylor

December 30, 2022 at 8:47 am

I am way behind the pace here because I got this as a Christmas gift and I am now working my way through. I'm having immense fun and learning a huge amount. Here is the code for a different game using the same sensors. It is a Formula 1 start time reaction game, for the F1

fans out there.

The game emulates the starting lights of a F1 race using the LEDs which light up until they are all lit. When the lights go out the race has started and you need to break the beam to record a time. I have even thrown in a poor impression of the starting engine sound :)

The code uses everything we have learned up to now with the addition of some functions from the time library to do accurate timing to the millisecond and also using the random number library to keep you guessing when the start lights will go out (clue – it will never be more than six seconds).

Given I knew nothing about microcontrollers and had never written a line of python before now, I am delighted with how much I have learned so quickly while having so much fun.

```
Here is the F1 start simulation game ##
from machine import Pin,PWM
import time, sys
import random # a random number generator set up detection beam
beam = Pin(26,Pin.IN,Pin.PULL_DOWN) set up buzzer and LEDs
buzzer = PWM)
buzzer.duty_u16(0)
buzzer.freq(500)
red_led = Pin(18,Pin.OUT)
amber_led = Pin(19,Pin.OUT)
green_led = Pin(20,Pin.OUT)

#simple function to clear the LEDs
def clear_leds():
red_led.value(0)
amber_led.value(0)
green_led.value(0)

Function to do a poor impression of a starting racing car
def racing_start():
for gear in range(4):
buzzer.duty_u16(300)
for i in range(400,800):
buzzer.freq(i)
time.sleep(0.0017*(gear+1))
buzzer.duty_u16(0)
time.sleep(0.15)
buzzer.duty_u16(0)
#initialise the main variables
start_time = 0
best_time = 0
best_time_set = False #Use this flag to see if a best time has been set yet

clear_leds() #Make sure all LEDs are off to start with
```

```
while True:

get a random number of seconds between “ready” and “go” in the range 1-6
holding_time = random.randint(1,6) count down to start using procession of LEDs at 1 second
intervals
red_led.value(1)
time.sleep(1)
amber_led.value(1)
time.sleep(1)
green_led.value(1) now hold the ‘driver’ for a random number of seconds
time.sleep(holding_time) #turn off the LEDs and get the time in microseconds clear_leds()
start_time = time.ticks_ms() wait for the beam break
while beam.value() == 1:
time.sleep(0.0000001) #Do nothing in this loop end_time = time.ticks_ms() # get the time in
milliseconds when the beam was broken reaction_time = time.ticks_diff(end_time,start_time)
#How long did the reaction take? print(“Reaction time was -”,reaction_time, “milliseconds”) if
not best_time_set: best_time = reaction_time best_time_set = True elif reaction_time <
best_time: best_time = reaction_time best_time_set = True print (“That was your best reaction
time so far!”) racing_start() time.sleep(5) # Wait for a while before starting the game again
```

I am way behind the pace here because I got this as a Christmas gift and I am now working my way through. I’m having immense fun and learning a huge amount. Here is the code for a different game using the same sensors. It is a Formula 1 start time reaction game, for the F1 fans out there.

The game emulates the starting lights of a F1 race using the LEDs which light up until they are all lit. When the lights go out the race has started and you need to break the beam to record a time. I have even thrown in a poor impression of the starting engine sound :)

The code uses everything we have learned up to now with the addition of some functions from the time library to do accurate timing to the millisecond and also using the random number library to keep you guessing when the start lights will go out (clue – it will never be more than six seconds).

Given I knew nothing about microcontrollers and had never written a line of python before now, I am delighted with how much I have learned so quickly while having so much fun.

Here is the F1 start simulation game ##

```
from machine import Pin,PWM
import time, sys
import random # a random number generator set up detection beam
beam = Pin(26,Pin.IN,Pin.PULL_DOWN) set up buzzer and LEDs
buzzer = PWM)
buzzer.duty_u16(0)
buzzer.freq(500)
red_led = Pin(18,Pin.OUT)
amber_led = Pin(19,Pin.OUT)
green_led = Pin(20,Pin.OUT)

#simple function to clear the LEDs
def clear_leds():
red_led.value(0)
amber_led.value(0)
```

```
green_led.value(0)

Function to do a poor impression of a starting racing car
def racing_start():
for gear in range(4):
buzzer.duty_u16(300)
for i in range(400,800):
buzzer.freq(i)
time.sleep(0.0017*(gear+1))
buzzer.duty_u16(0)
time.sleep(0.15)
buzzer.duty_u16(0)
#initialise the main variables
start_time = 0
best_time = 0
best_time_set = False #Use this flag to see if a best time has been set yet

clear_leds() #Make sure all LEDs are off to start with

while True:

get a random number of seconds between “ready” and “go” in the range 1-6
holding_time = random.randint(1,6) count down to start using procession of LEDs at 1 second intervals
red_led.value(1)
time.sleep(1)
amber_led.value(1)
time.sleep(1)
green_led.value(1) now hold the ‘driver’ for a random number of seconds
time.sleep(holding_time) #turn off the LEDs and get the time in microseconds clear_leds() start_time =
time.ticks_ms() wait for the beam break
while beam.value() == 1:
time.sleep(0.0000001) #Do nothing in this loop end_time = time.ticks_ms() # get the time in
milliseconds when the beam was broken reaction_time = time.ticks_diff(end_time,start_time) #How
long did the reaction take? print(“Reaction time was -”,reaction_time, “milliseconds”) if not
best_time_set: best_time = reaction_time best_time_set = True elif reaction_time < best_time: best_time
= reaction_time best_time_set = True print (“That was your best reaction time so far!”) racing_start()
time.sleep(5) # Wait for a while before starting the game again
```

Jamie

December 30, 2022 at 8:47 am

Had so much fun with this box (we are quite far behind!) We all have sore fingers now but overly competitive mum is determined to beat uncles score!

Had so much fun with this box (we are quite far behind!) We all have sore fingers now but overly competitive mum is determined to beat uncles score!

The Pi Hut

December 28, 2022 at 1:30 pm

@Victoria – The Box Contents section of each page mentions the resistor rating required for each day's components, and you can identify what your resistors are by checking their colour bands and using an online resistor colour code calculator such as https://www.digikey.com /en/resources/conversion-calculators/conversion-calculator-resistor-color-code (You can also narrow this down further by seeing how many of each you have, as for example, on day #2 you get 3x 330-ohm resistors…but you should only have 1x 4.7K ohm resistor)

@Victoria – The Box Contents section of each page mentions the resistor rating required for each day's components, and you can identify what your resistors are by checking their colour bands and using an online resistor colour code calculator such as https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code (You can also narrow this down further by seeing how many of each you have, as for example, on day #2 you get 3x 330-ohm resistors…but you should only have 1x 4.7K ohm resistor)

Victoria

December 28, 2022 at 1:08 pm

I would appreciate documentation about what resistor strength is appropriate for each component. I the resisters mixed up.

I would appreciate documentation about what resistor strength is appropriate for each component. I the resisters mixed up.

The Pi Hut

December 23, 2022 at 8:43 am

@Rafe – Anything like this (using IR) can be a little unreliable when used outside or in direct sunlight, as sunlight itself includes IR waves which can confuse the sensor somewhat. One solution is to fabricate a cover or partial enclosure to shield your project from sunlight as much as possible. Another option (if you're looking to make your own project) is to use a more suitable sensor depending on where it's going to live and what it's used for. For example, an ultrasonic distance sensor could be a good alternative to detecting something blocking a path: https://thepihut.com/products/ultrasonic-distance-sensor-hcsr04

@Rafe – Anything like this (using IR) can be a little unreliable when used outside or in direct sunlight, as sunlight itself includes IR waves which can confuse the sensor somewhat. One solution is to fabricate a cover or partial enclosure to shield your project from sunlight as much as possible. Another option (if you're looking to make your own project) is to use a more suitable sensor depending on where it's going to live and what it's used for. For example, an ultrasonic distance sensor could be a good alternative to detecting something blocking a path: https://thepihut.com/products/ultrasonic-distance-sensor-hcsr04

Markus

December 23, 2022 at 8:44 am

Sorry, to get back on the 10K resistor.
As you explained, it is added as an external pull up resistor, so that the beam can pull the signal low.
So I would expect to have the GPO Pin setup with beam = Pin(26), simply without any pullup/down flag.
But the code sets it up with Pin(26, PULL_DOWN)? Why is this? Shouldn't it be Pin(26, PULL_UP) when omitting the external resistor and just Pin(26), when the resistor is in place?

Sorry, to get back on the 10K resistor.
As you explained, it is added as an external pull up resistor, so that the beam can pull the signal low.
So I would expect to have the GPO Pin setup with beam = Pin(26), simply without any pullup/down flag.
But the code sets it up with Pin(26, PULL_DOWN)? Why is this? Shouldn't it be Pin(26, PULL_UP) when omitting the external resistor and just Pin(26), when the resistor is in place?

Rafe

December 23, 2022 at 8:39 am

I got the same result as @Alex, the sensor worked fine in the evening but in the day the daylight interfered with the sensor making it almost impossible to break the beam. Is there any way to fix this?

I got the same result as @Alex, the sensor worked fine in the evening but in the day the daylight interfered with the sensor making it almost impossible to break the beam. Is there any way to fix this?

The Pi Hut

December 19, 2022 at 11:04 am

Best to remove everything and start again, but if you've already tried that and still no joy, pop us a quick message via support.thepihut.com we'll get a replacement sensor sent out to you

Best to remove everything and start again, but if you've already tried that and still no joy, pop us a quick message via support.thepihut.com we'll get a replacement sensor sent out to you

Jay

December 19, 2022 at 11:01 am

I can't seem to get past exercise 1, the sensor is constantly reading as 0/broken, even with me checking and re-checking all connections, and moving the two sensor parts very close to each other.

Any ideas what I might focus on re-checking? Or a way to make sure that the sensor is not malfunctioning?

I can't seem to get past exercise 1, the sensor is constantly reading as 0/broken, even with me checking and re-checking all connections, and moving the two sensor parts very close to each other.

Any ideas what I might focus on re-checking? Or a way to make sure that the sensor is not malfunctioning?

The Pi Hut

December 12, 2022 at 11:13 am

@Markus This is to pull 'up' the sensor. Technically you might be able to do this using pull-ups on the pico in your code, however we went down the physical pull-up route with this box.

@Markus This is to pull 'up' the sensor. Technically you might be able to do this using pull-ups on the pico in your code, however we went down the physical pull-up route with this box.

The Pi Hut

December 12, 2022 at 11:01 am

@Steve – Swiping will lead to an instant disqualification ;)

Good idea on the game time variable, hopefully by this stage in the calendar our budding makers will have the confidence and knowledge to tweak our examples and improve them like that.

@Steve – Swiping will lead to an instant disqualification ;)

Good idea on the game time variable, hopefully by this stage in the calendar our budding makers will have the confidence and knowledge to tweak our examples and improve them like that.

The Pi Hut

December 12, 2022 at 10:57 am

@Gordon – You may well be this year's champ! Nice work!

@Gordon – You may well be this year's champ! Nice work!

Steve

December 12, 2022 at 10:56 am

197 tapping, does swiping on the desk count? I got 323 swiping :p lol. That's as much as my not so well exercised late 30s arm is going to do, it soon makes your arm ache.

Honestly going in I was thinking this would be a quick day being another sensor with just a HIGH/LOW response. It also didn't help that my emitter had a broken red wire. Though I got the plastic case open and used a crocodile clip and a small bit of solid jumper wire to get it attached to the PCB inside and onto the breadboard, it worked fine :).

> But I did have some fun with this and I think the longer code examples will be good for beginners though I think you should have had the 30 seconds value in a variable not hard coded in multiple places :p :).

197 tapping, does swiping on the desk count? I got 323 swiping :p lol. That's as much as my not so well exercised late 30s arm is going to do, it soon makes your arm ache.

Honestly going in I was thinking this would be a quick day being another sensor with just a HIGH/LOW response. It also didn't help that my emitter had a broken red wire. Though I got the plastic case open and used a crocodile clip and a small bit of solid jumper wire to get it attached to the PCB inside and onto the breadboard, it worked fine :).

But I did have some fun with this and I think the longer code examples will be good for beginners though I think you should have had the 30 seconds value in a variable not hard coded in multiple places :p :).

Markus

December 12, 2022 at 11:11 am

> Why do we need the 10K resistor between red and white?

Why do we need the 10K resistor between red and white?

Alex

December 12, 2022 at 11:16 am

> I struggled with this sensor. The beam only reported being broken when the sensor was almost completely covered. Figured out it was being triggered by the light in the room. (Perhaps a burglar alarm or daylight flooding it?). So we built a small enclosure for it, and had better success. Wondered if the resistor isn't allowing it to be sensitive enough.

I struggled with this sensor. The beam only reported being broken when the sensor was almost completely covered. Figured out it was being triggered by the light in the room. (Perhaps a burglar alarm or daylight flooding it?). So we built a small enclosure for it, and had better success. Wondered if the resistor isn't allowing it to be sensitive enough.

Gordon

December 10, 2022 at 10:00 pm

> 248 …Ouch!!

248 …Ouch!!

Leave a comment

All comments are moderated before being published.

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Name

E-mail

Message

SUBMIT

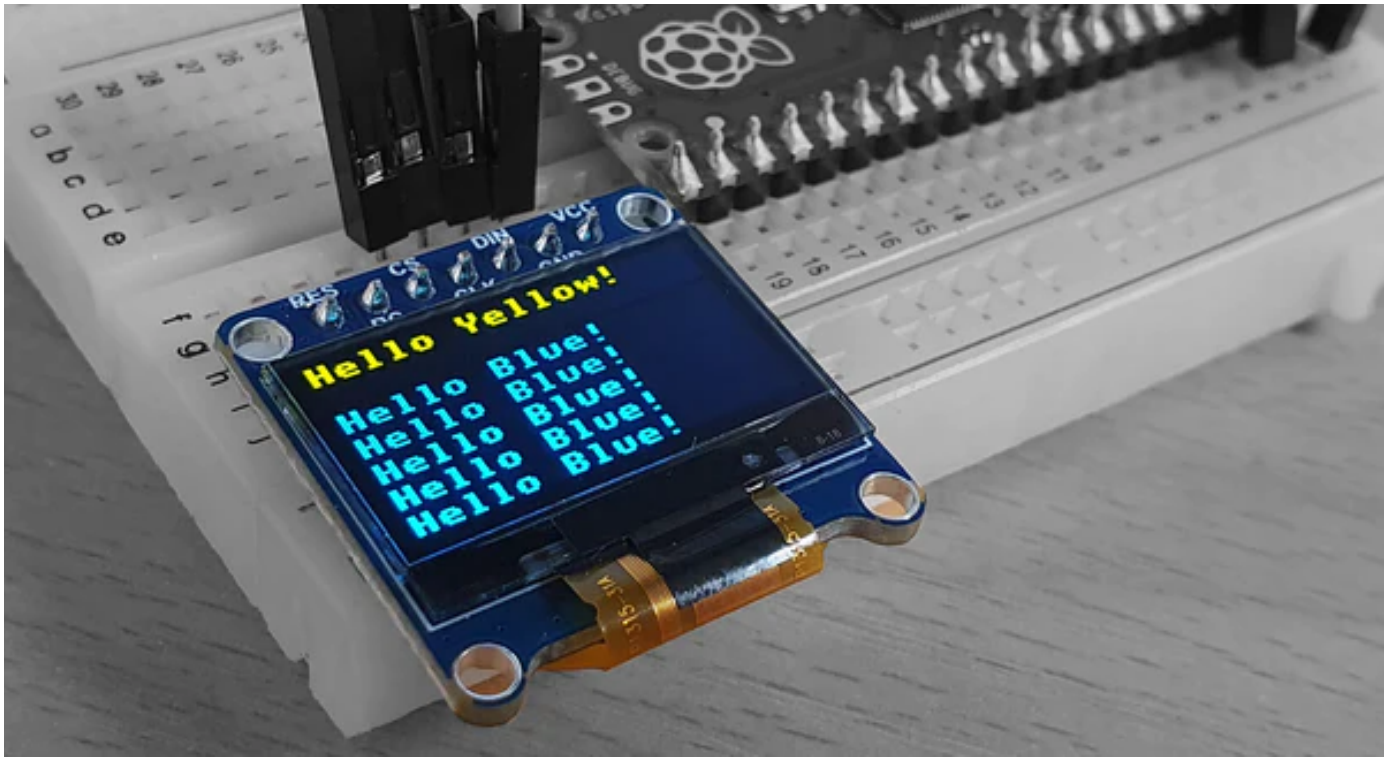
Related Posts

MicroPython Skill Builders - #2 LEDs & Lists

Tony Goodhew • May 30, 2023

This is the second instalment of our MicroPython Skill Builders series by Tony Goodhew, aiming to improve your coding skills with MicroPython whilst introducing new components and coding techniques - using...

Read more

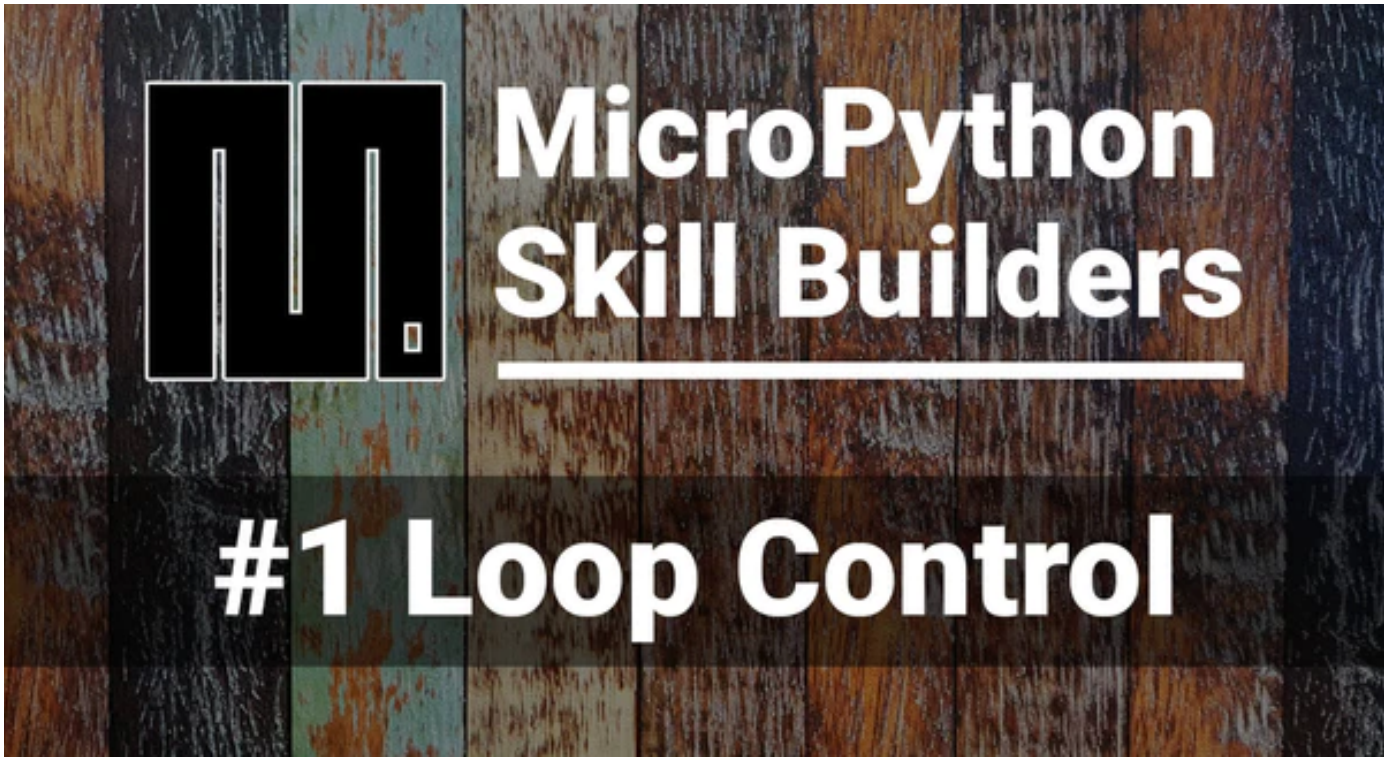


Using an SSD1315 OLED with the Raspberry Pi Pico

The Pi Hut • May 26, 2023

A couple of weeks back we started stocking an interesting new OLED display from our friends over at Waveshare - a two-colour SSD1315 0.96" OLED to be exact. Funky dual-colours...

Read more



MicroPython Skill Builders - #1 Smarter Loops

Tony Goodhew • May 9, 2023

In this new MicroPython Skill Builders series by Tony Goodhew, we aim to help improve your coding skills in MicroPython whilst introducing new components and coding techniques - all using...

Read more

Handy Links

All Products

FAQs

Popular Searches

Search

Site Reviews

Got any questions?

Contact Us / Support Portal

Can I Cancel My Order?

Has My Order Shipped Yet?

Where Is My Order?

Do You Ship To {insert country name}

How Much Is Shipping?

Terms & Conditions

Delivery

Lithium Shipping

Pre-Orders

Privacy Statement

Policies

Terms of Service

Company Info

FAQ

Klarna FAQ

Quick Start Guide

Search

Support Portal

Our Store Sections

Raspberry Pi

Maker Store

micro:bit

Arduino

Gifts

Follow us

Raspberry Pi
Approved Reseller

We accept