Binary Inputs

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #6

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

Each GPIO pin can be

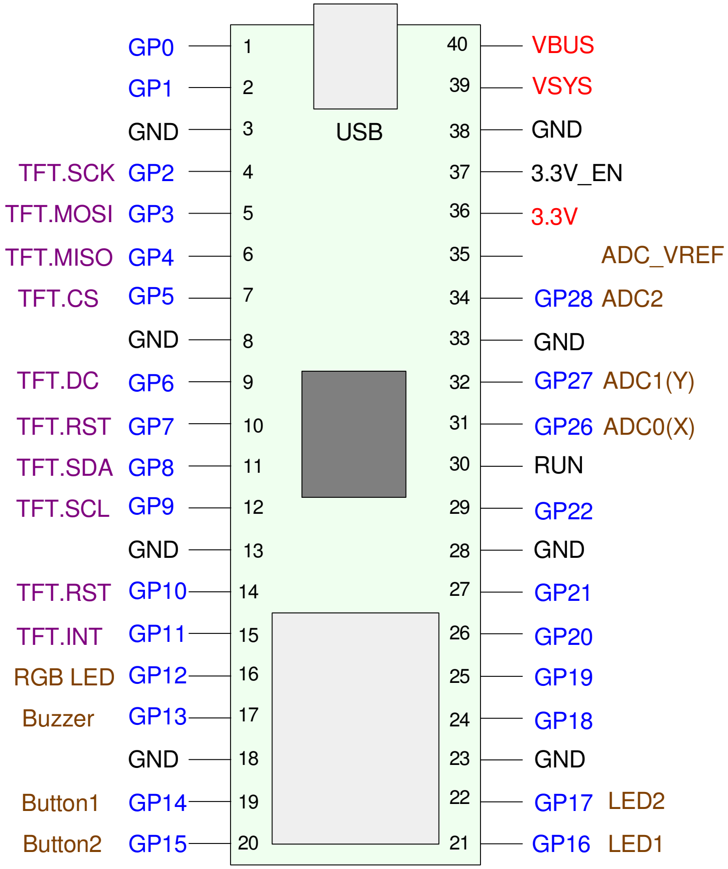
- Binary Outputs (last lecture), or
- Binary Inputs (this lecture)

as well as other functions (coming later).

Similar to our last lecture

- 0V is read as logic 0
- 3.3V is read as logic 1

Do not apply 5V to the GP pins
Doing so might destroy the Pico board.



This lecture looks at

- Converting push buttons to binary (0V & 3.3V) logic levels
- Reading a numeric keypad
- Converting voltages, resistance's, and temperatures to 0V / 3.3V logic levels,
- Counting edges & building a voting machine
- Measuring a button's on-time and building a debate moderator
- Counting multiple edges and writing a Hungry-Hungry Hippo game.

Reading Push Buttons:

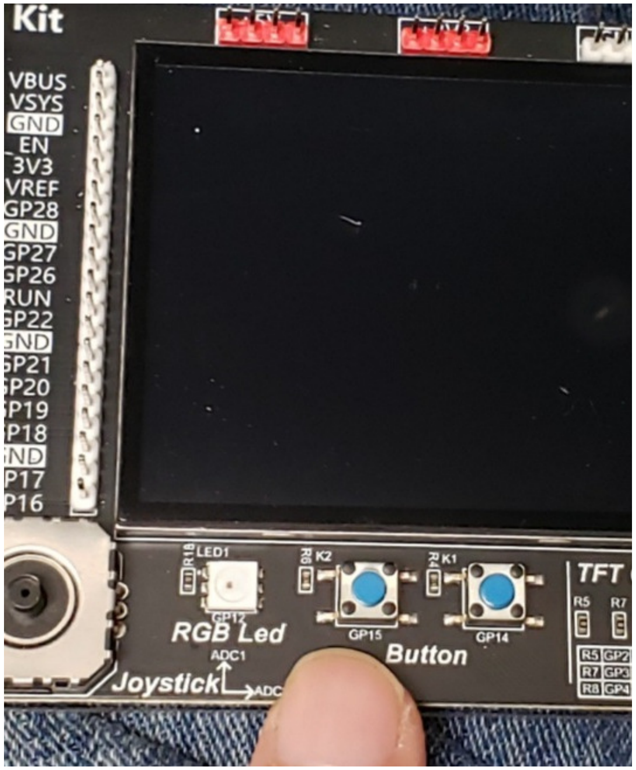
The Pi-Pico Breadboard has two push buttons

- GP15
- GP14

To read the buttons, these need to be inputs:

Three options exist:

```
from machine import Pin

Button = Pin(15, Pin.IN)
Button = Pin(15, Pin.IN, Pin.PULL_UP)
Button = Pin(15, Pin.IN, Pin.PULL_DOWN)
```



Button = Pin(15, Pin.IN)

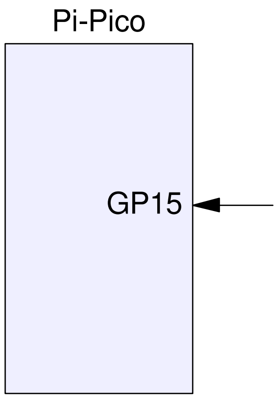
- Pin 15 is input and floating
- Hardware is responsible for setting the voltage to 0V or 3.3V

Button = Pin(15, Pin.IN,Pin.PULL_UP)

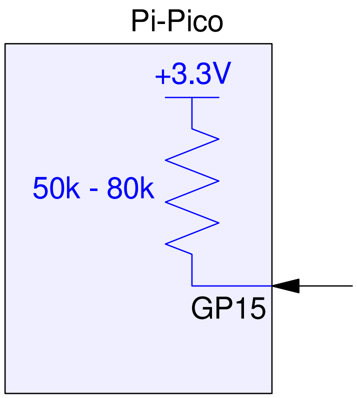
- A 50-80k resistor ties pin 15 to +3.3V

Button = Pin(15, Pin.IN,Pin.PULL_DOWN)

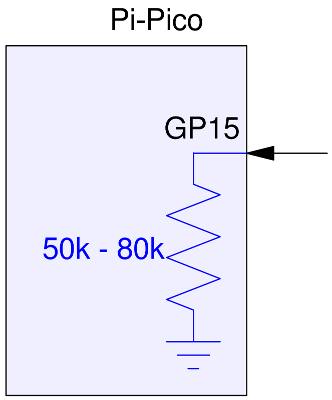
- A 50-80k resistor ties pin 15 to +0V



none



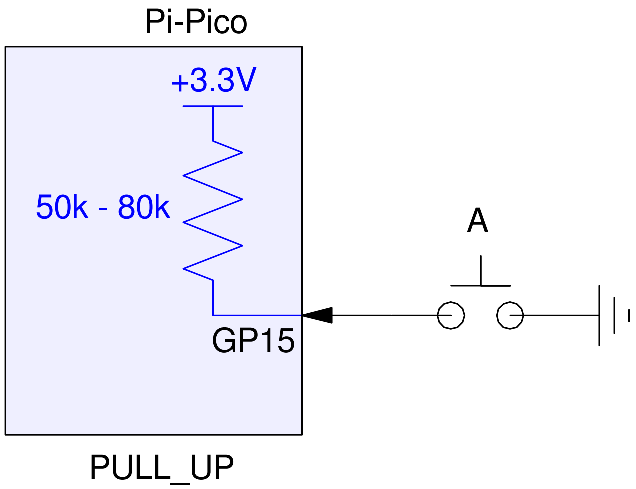
PULL_UP

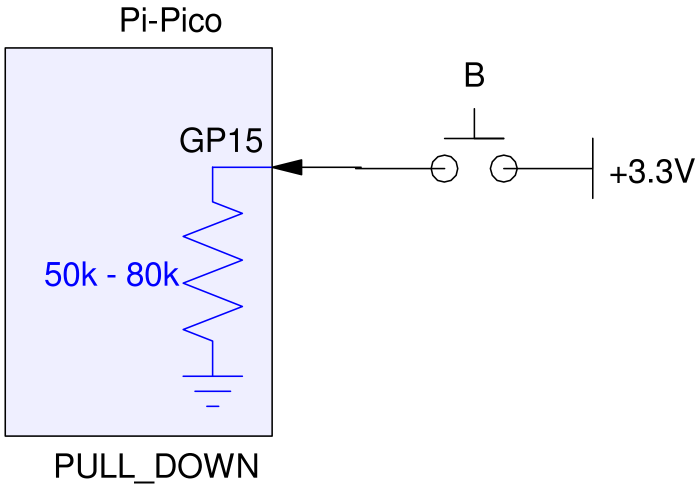


PULL_DOWN

Both pull-up and pull-down can be used along with a momentary switch to read if the switch is pressed or not:

- Pull-Up: GP15 is logic 1 if A is not pressed and 0 if A is pressed
- Pull-Down: GP15 is logic 1 if A is pressed and 0 if A is not pressed





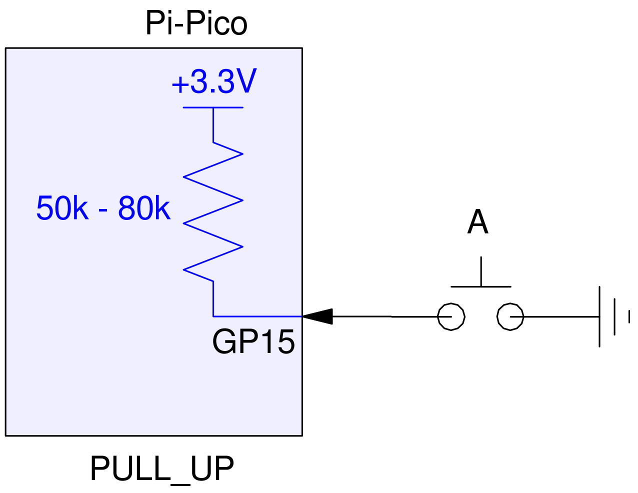
In general, the pull-up setting is safest

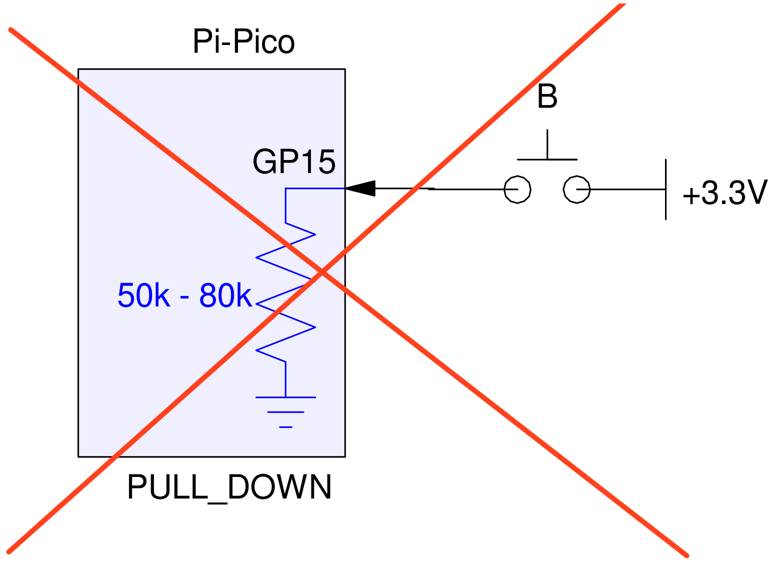
- Pushing the button will not damage the Pico chip - you're just connecting it to ground

The pull-down setting can damage your Pico board:

- If you accidentally use +5V rather than +3.3V, pressing the button will fry your Pico board

Stick with the pull-up option with the switch tied to ground.





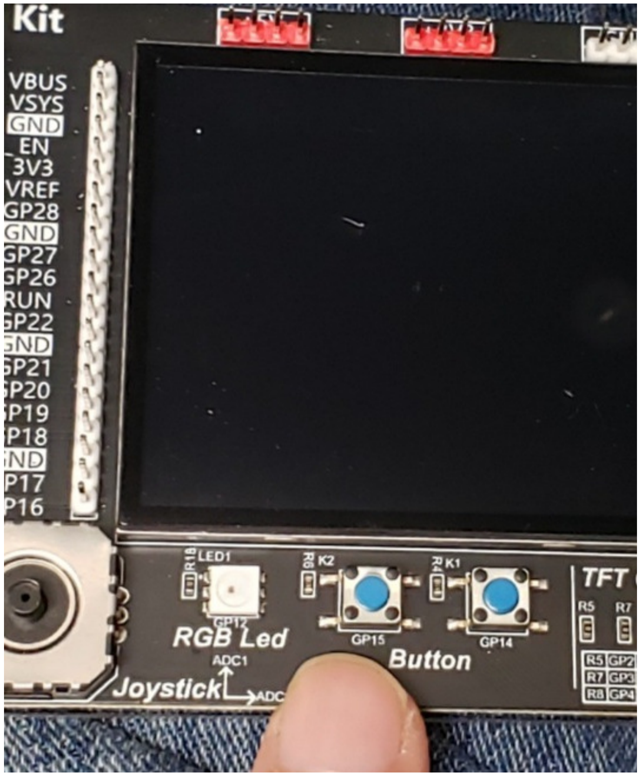
Sample Code: The following program displays

- 1 when button 15 is not pressed
- 0 when button 15 is pressed

```
from machine import Pin
from time import sleep_ms

Button = Pin(15, Pin.IN, Pin.PULL_UP)

while(1):
    X = Button.value()
    print(X)
    sleep_ms(100)
```

shell

```
0
0
0
1
1
1
0
0
0
```



More Buttons!

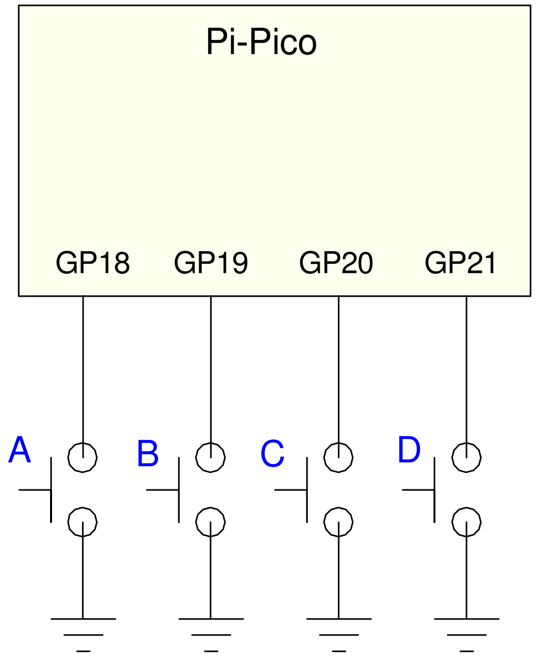
- When two buttons are not enough...

If you need more buttons, simply

- Attach each GPIO pin to ground through the button
- Set each pin to input
- Using a pull-up resistor

This uses up a lot of I/O pins, however

```
from machine import Pin

A = Pin(18, Pin.IN, Pin.PULL_UP)
B = Pin(19, Pin.IN, Pin.PULL_UP)
C = Pin(20, Pin.IN, Pin.PULL_UP)
D = Pin(21, Pin.IN, Pin.PULL_UP)
```



Sidelight: Boolean Logic with Momentary Switches

NOT:

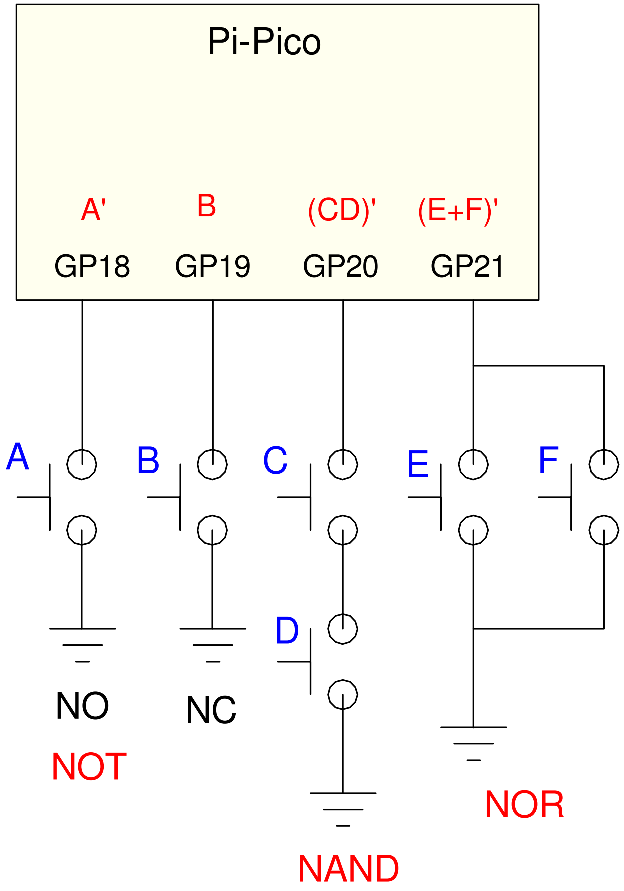
- Using a normally-open (NO) switch results in a NOT function
  - Normally-closed (NC) results in Y=X

NAND:

- Place switches in series
- Pressing both switches results in logic 0

NOR

- Place switches in parallel
- Pressing either switch results in logic 0

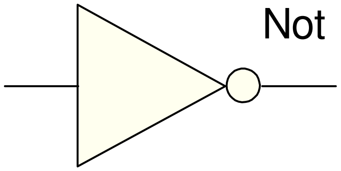


Anything you can do in hardware you can do in software

- and visa versa

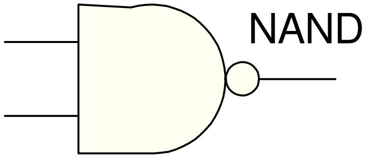
You can also buy

- Normally Open Switches
- Normally Closed Switches
- These let you implenent A' and B' in hardware

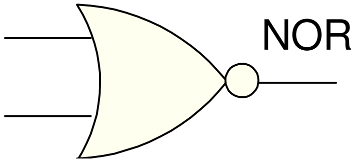


With NOT, NAND, and NOR,

- You can implement logic funcitons in hardware
- You can also implement these in software



It's you choice as the design engineer which you use



Numeric Keypad (Hardware)

- Another way to input data to a Pico

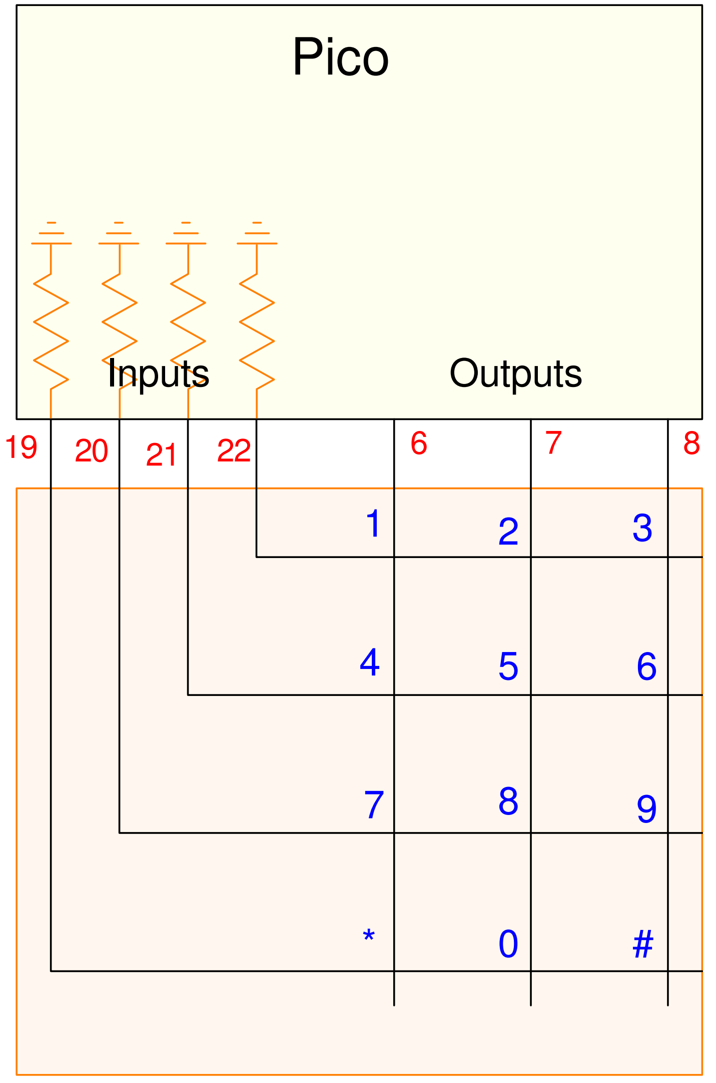
Connect the rows and colums to a Pi-Pico

Set the columns to output

- 0V or 3.3V

Set the rows to input

- Use internal pull-down resistors



Numeric Keypad

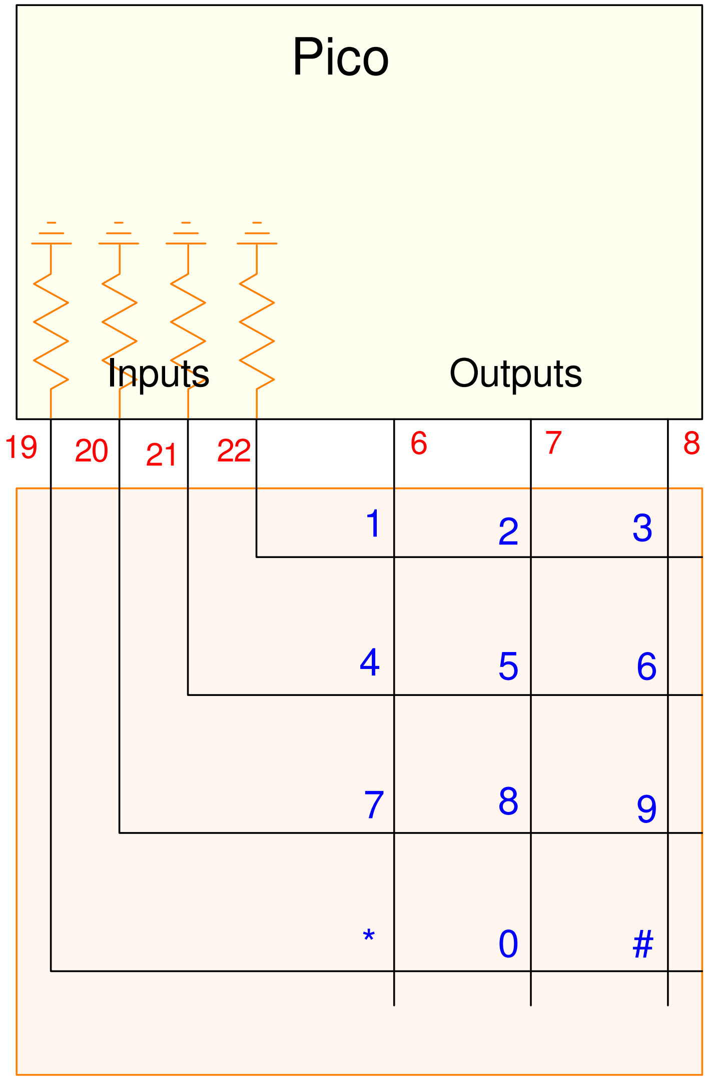
Numeric Keypad (Software)

def GetKey()

- Scan column #1 (6/7/8 = 1/0/0)
  - scan column #1
  - if GP22, key = 1
  - if GP21, key = 4
  - if GP20, key = 7
  - if GP19, key = 10 (*)
- Repeat for column 2 and 3
- Return key pressed (255 = no key)

def ReadKey()

- Call GetKey()
- Wait until a key is pressed
  - returned value if not 255
- Wait until key is released
  - return value = 255
- Return key value



Numeric Keypad

Numeric Keypad Demo

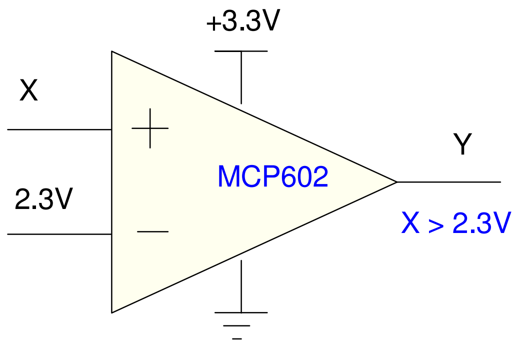
- YouTube

Reading Voltage

- X > 2.3V

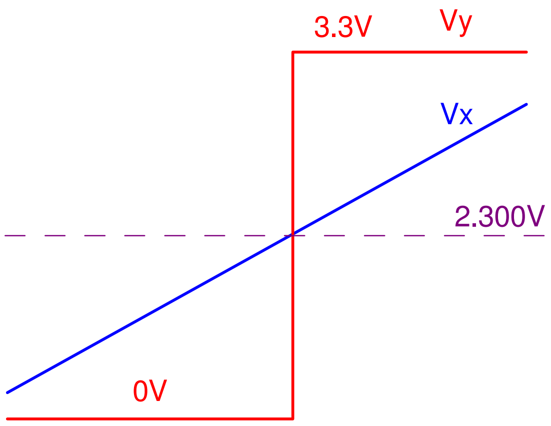
Use a comparitor (MCP602 op-amp works)

- Output 3.3V when X > 2.3V
- Output 0V when X < 2.3V



Note: the op-amp used needs

- To opeate from a single power supply
- To operate over a 0V - 3.3V range
- Rail-to-rail outputs



An MCP602 does this

- LM741 or LM833 do not.

Reading Resistance:

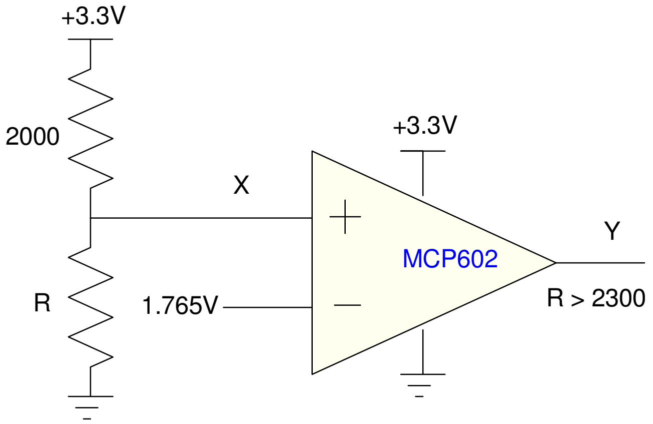
- R > 2300 Ohms

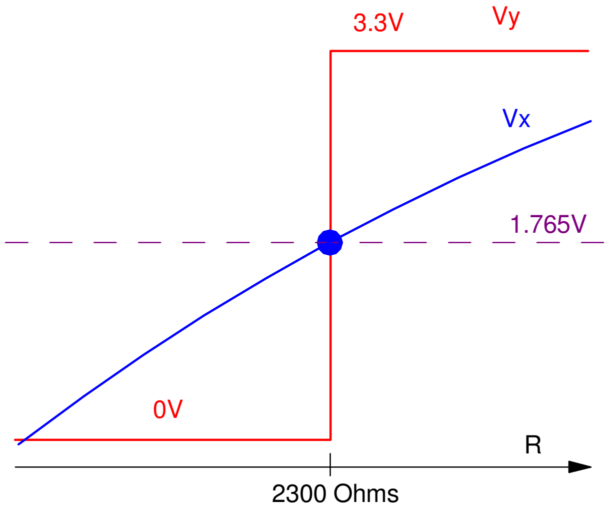
Trick: Change the problem

- Convert resistance to a voltage
- Use the previous circuit

Example:

- Use a voltage divider
- With a 2k resitor
- At R = 2300 Ohms
- $X = \left(\frac{R}{R+2000}\right)3.3V = 1.765V$
- Switch at 1.765 Volts



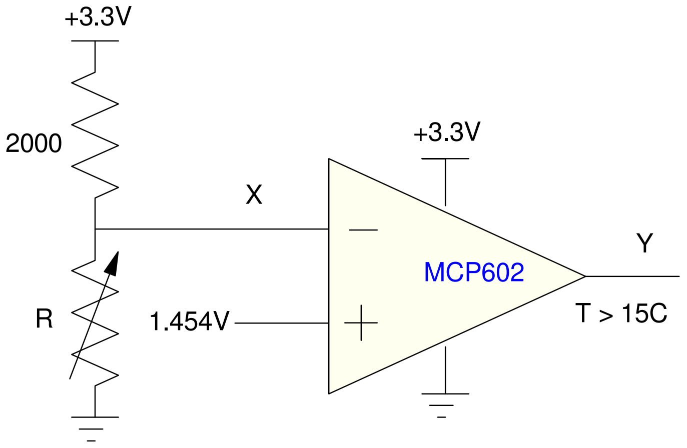


Reading Temperature:

- T > 15C

Trick

- Convert temperature to resistance
- Find the R(15C), then
- Use the previous circuit



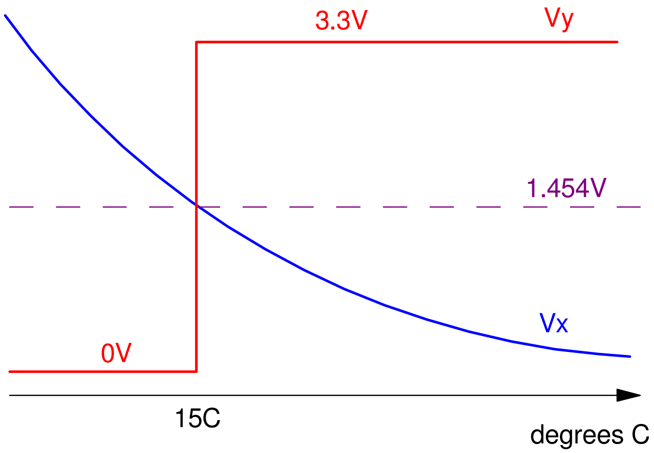
Example: Output 3.3V for T > 15C

Pick a thermistor, such as

$$R = 1000 \cdot \exp\left(\frac{3905}{T+273} - \frac{3905}{298}\right)\Omega$$

At 15C

- R = 1576 Ohms
- Vx = 1.454V

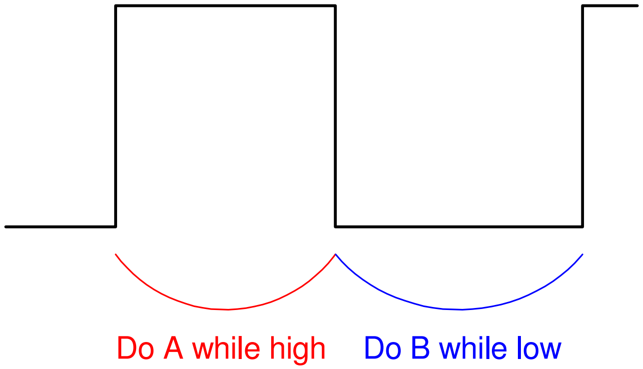


Level vs. Edge-Sensitive Programs

Once you can read the input

- button
- voltage
- temperature

Have that input affect the program

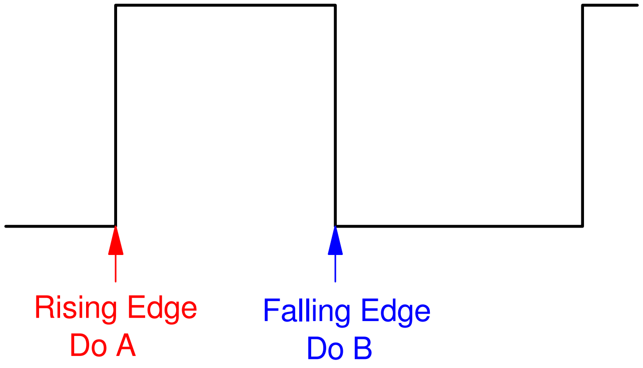


Level Sensitive Programs

- Operation depends upon the logic level

Edge Sensitive Programs

- Operations happen on rising and falling edges



Level Sensitive: Debate Moderator

- Prevent candidates from talking over each other

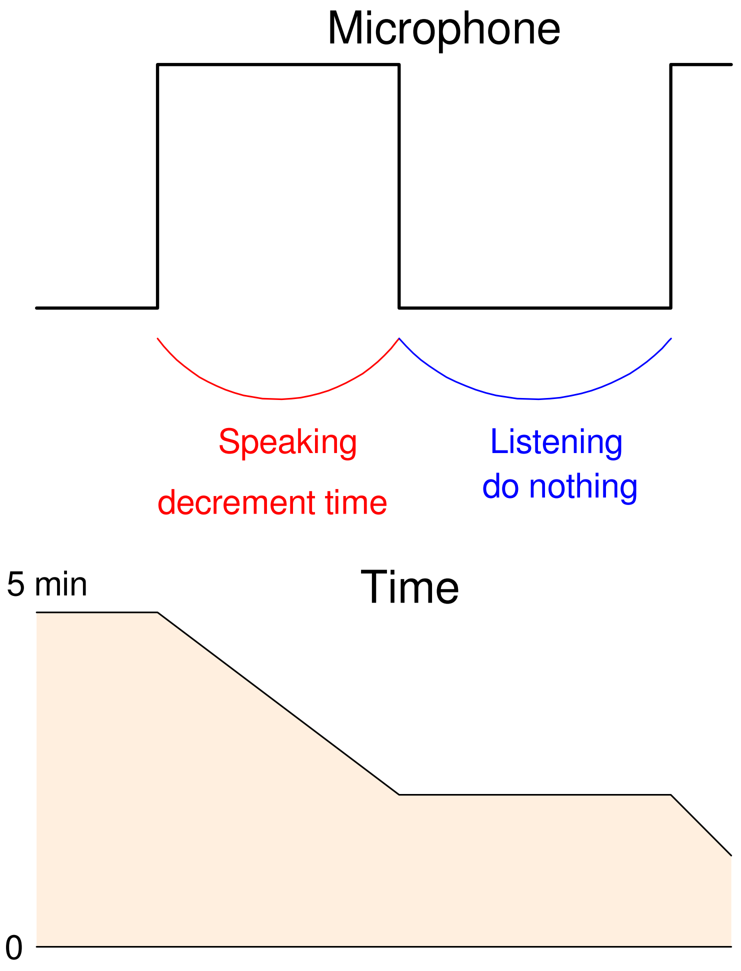
Connect a microphone to a binary input

- 3.3V: Candidate is speaking
- 0V: Candidate is listenting

Initially, each candidate is given 5 min

- When you speak, your clock runs down
- When silent, your clock remains constant

When your time reaches zero, your microphone cuts off



Debate Moderator: Software

- Test code using push buttons
- Button press (0V) = speaking

Each candidate is given 5 minutes

- 300 seconds

Every 100ms

- Check each microphone
- If speaking, decrement their time

When you reach zero

- Turn off the microphone
- (not in code)

```
# Debate Moderator
from machine import Pin
from time import sleep_ms

ButtonA = Pin(15, Pin.IN, Pin.PULL_UP)
ButtonB = Pin(14, Pin.IN, Pin.PULL_UP)

ATime = 300.0
BTime = 300.0

while(1):
    if(ButtonA.value() == 0):
        if(ATime > 0):
            ATime -= 0.1
    if(ButtonB.value() == 0):
        if(BTime > 0):
            BTime -= 0.1
    print(ATime, BTime)
    sleep_ms(100)
```

Debate Moderator: Hardware

Stage 1: Microphone

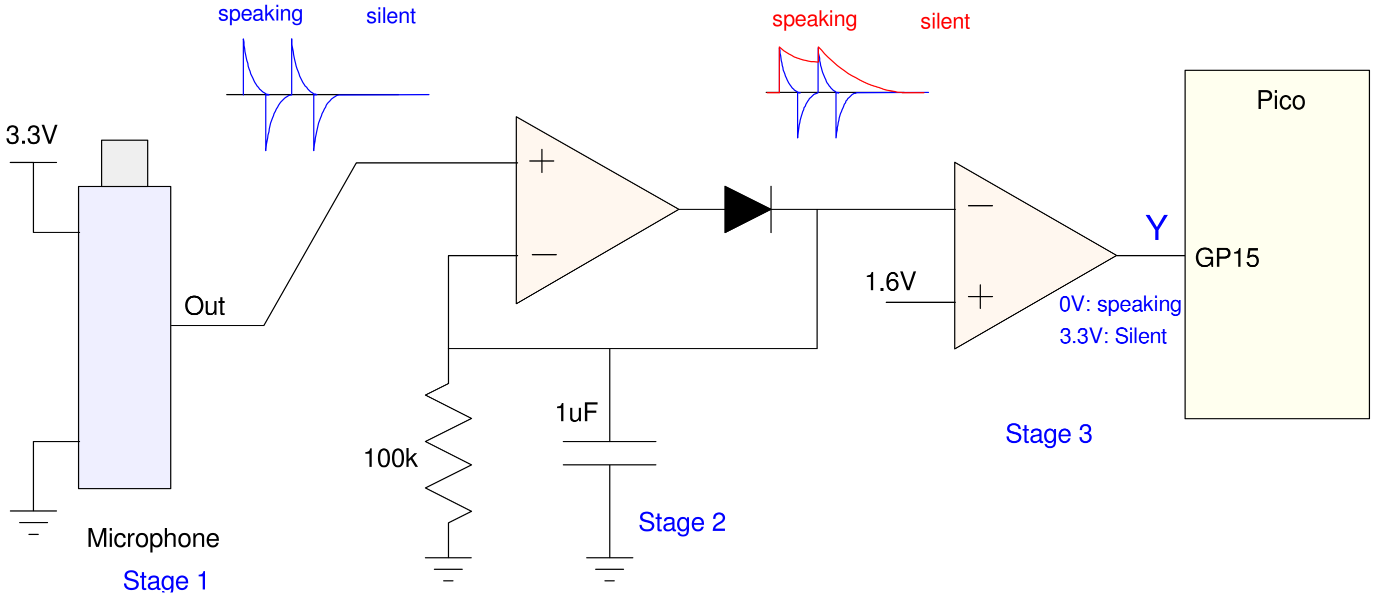
- Daoki high sensitivity microphone (Amazon $1.20)

Stage 2: Envelope Detector

- Hold peaks

Stage 3: Comparitor

- Output 0V or 3.3V



Debate Moderator Demo

- YouTube

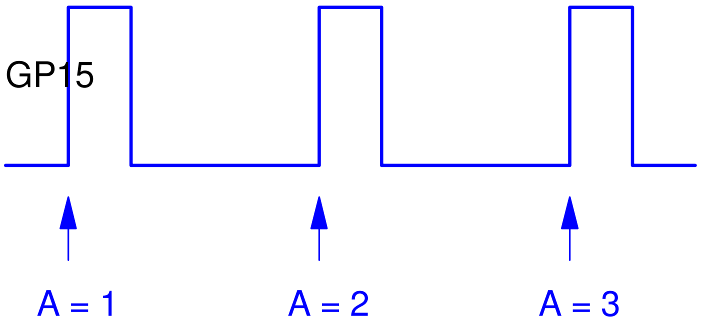
Edge Sensitive Program: Voting Machine

A second type of program counts edges

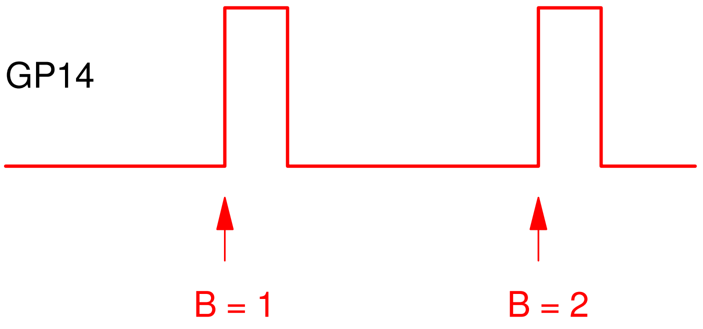
- Action only takes place during the rising edge and/or falling edge of a signal

Example: Voting Machine

- Count rising edges on GP15
    - Candidate A
- Count rising edges on GP14
    - Candidate B



Candidate A



Candiate B

Voting Machine with One Candidate

Use two wait-loops

- Wait until button is pressed
  - Button goes to 0
- Wait until button is released
  - Button goes to 1

The rising edge has been detected

- Add one vote (one count)

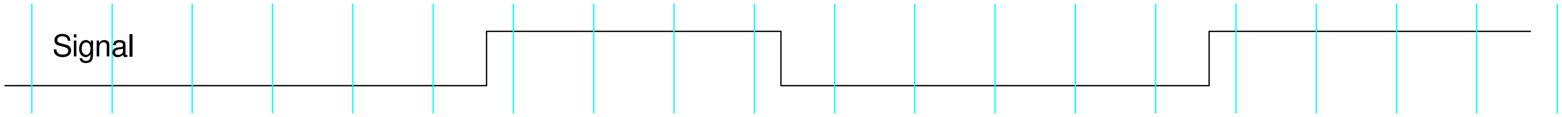
```
from machine import Pin
from time import sleep_ms

Button = Pin(15, Pin.IN, Pin.PULL_UP)

Count = 0
print('Press and release button to count')
while(1):
    while(Button.value() == 1):
        pass
    while(Button.value() == 0):
        pass
    Count += 1
    print(Count)
```

Voting Machine with Two Candiates

Look for a 0 to 1 transition

- If the current reading is a 1, and
- The previous reading was a 0

you just detected a rising edge.



Current Reading	0	0	0	**1**	1	1	1	0	0	0	0	0	**1**	1	1	1
Previous Reading		0	0	**0**	1	1	1	1	0	0	0	0	**0**	1	1	1
				rising edge									rising edge			

A rising edge is detected when the current signal is 1 and its previous value was 0

Code:

Vote for A if

- Current value is 1 and
- Previous value was 0

Vote for B if

- Current value is 1 and
- Previous value was 0

```
# Voting Machine
# input 14 and 15

from machine import Pin
from time import sleep_ms

PlayerA = Pin(15, Pin.IN, Pin.PULL_UP)
PlayerB = Pin(14, Pin.IN, Pin.PULL_UP)

A = 1
B = 1
Na = 0
Nb = 0
time = 0
while(1):
    zA = A
    A = PlayerA.value()
    zB = B
    B = PlayerB.value()
    if( (A==1) & (zA==0) ):
        Na += 1
    if( (B==1) & (zB==0) ):
        Nb += 1
    print('Votes for A ',Na, '   Votes for B ',Nb)
    sleep_ms(100)
```

Hungry-Hungry Hippo

https://youtu.be/Rf3ow_DdmtE?feature=shared

Finally, let's use the push buttons to play a game of *Hungry-Hungry Hippo*

- Each player starts with 10.00 seconds
- Each player presses their button as fast as they can, with each button release (rising edge) tallied
- Once 10 seconds is over, the game is over.

This is similar to a voting machine, except

- The time is limited to 10 seconds.
- Once time is over, stop counting.
- Sample every 10ms so you don't miss points

Flags

This program uses a flag

- Flags indicate something happened
- Such as a button press

The score is only updated on scores

- rather than every 10ms
- as indicated by flag==1
- makes the display prettier

```
from machine import Pin
from time import sleep_ms

PlayerA = Pin(15, Pin.IN, Pin.PULL_UP)
PlayerB = Pin(14, Pin.IN, Pin.PULL_UP)

A = B = 1
Na = Nb = time = flag = 0
print('Press buttons to count')

while(time < 10):
    zA = A
    A = PlayerA.value()
    zB = B
    B = PlayerB.value()
    if( (A==1) & (zA==0) ):
        Na += 1
        flag = 1
    if( (B==1) & (zB==0) ):
        Nb += 1
        flag = 1
    if(flag == 1):
        print(Na, Nb)
        flag = 0
    sleep_ms(10)
    time += 0.01

print('Game Over')
if(Na > Nb):
    print('Player A Wins')
elif(Nb > Na):
    print('Player B Wins')
else:
    print('Tie')
```

Summary

Each I/O pin can be set up as a binary input or binary output. For binary inputs

- 0V is read as logic 0,
- 3.3V is read as logic 1, and
- 5V may destroy your Pico board (don't do it)

These inputs can control a program's flow

- Using the level of the signal (logic 1 or 0), or
- Using the edges of the signal (rising or falling)

References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- https://learn.pimoroni.com/article/getting-started-with-pico
- https://www.w3schools.com/python/default.asp
- https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
- https://docs.micropython.org/en/latest/library/index.html
- https://www.fredscave.com/02-about.html

Pi-Pico Breadboard Kit

- https://wiki.52pi.com/index.php?title=EP-0172

Other

- https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/
- https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/
- https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/