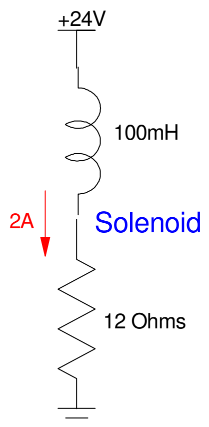
ECE 476/676 - Test #1: Name ____________________

1) **Hardware: Binary Output** A solenoid (electronic door-lock - modeled as a 100mH inductor and a 12 Ohm resistor) requires 2A at 24VDC to turn on. Design a circuit so that a Pi-Pico can turn on and off this solenoid using one of its binary outputs. Note that the output of a Pi-Pico is

- Vout = 0V or 3.3V
- Iout < 12mA

If you need to make assumptions about the hardware you are using, state the assumptions you're making



Assume a NPN transistor (2SC144) with

- $\beta = 200$
- $V_{ce(sat)} = 0.36V$
- $I_c(\max) = 10A$

To saturate the transistor, we need

$$\beta I_b > I_c$$

$$I_b > \left(\frac{2A}{200}\right) = 10mA$$

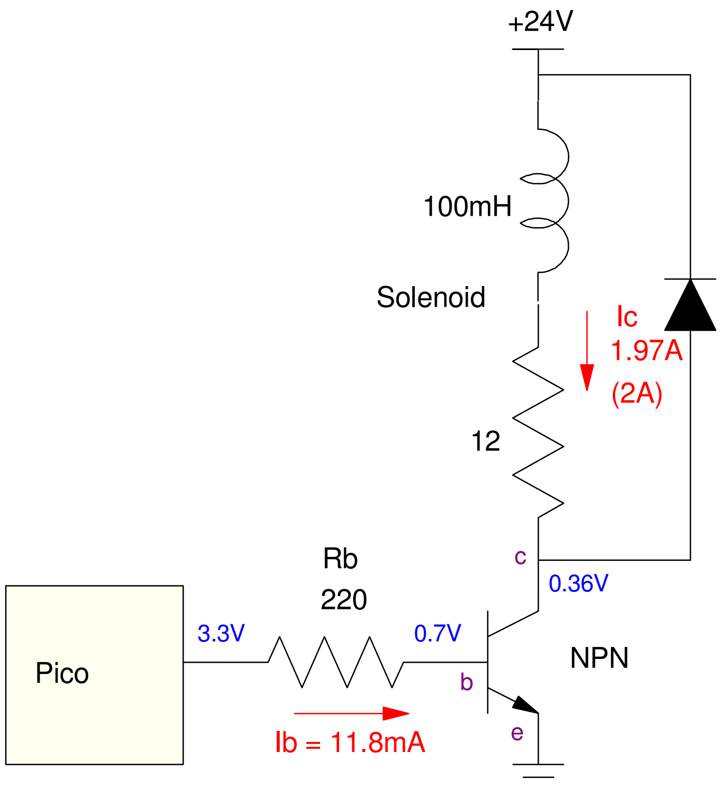
Pick a number between 10mA and 12mA (max a Pico can output). Let Ib = 12mA (max current)

$$R_b = \left(\frac{3.3V-0.7V}{12mA}\right) = 216\Omega$$

Let Rb = 220 Ohms (results in 11.8mA)

Add a flyback diode to save the transistor when you turn off the solenoid. (clips Vc at +24.7V)

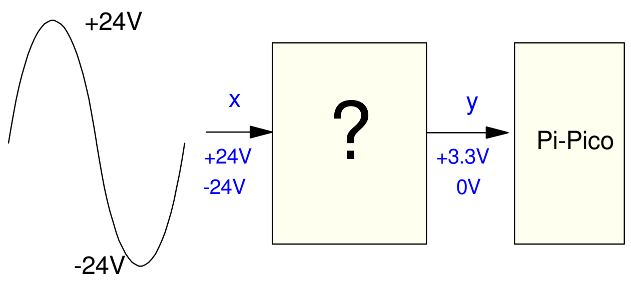
There are other solutions as well...



2) Hardware: Analog Inputs Design a circuit to allow a Pi-Pico to read an analog input which can vary from -24V to +24V

- -24V in produces 0V out
- +24V in produces +3.3V out
- Proportional inbeteen

$$y = \left(\frac{3.3}{48}\right)(x+24) = \left(\frac{3.3}{48}\right)x + 1.65$$



Write this as

$$y = 0.06875(x) + 0.5(3.3V) + 0.43125(0V)$$

(make all the coefficients add to 1.000 for a weighted average).

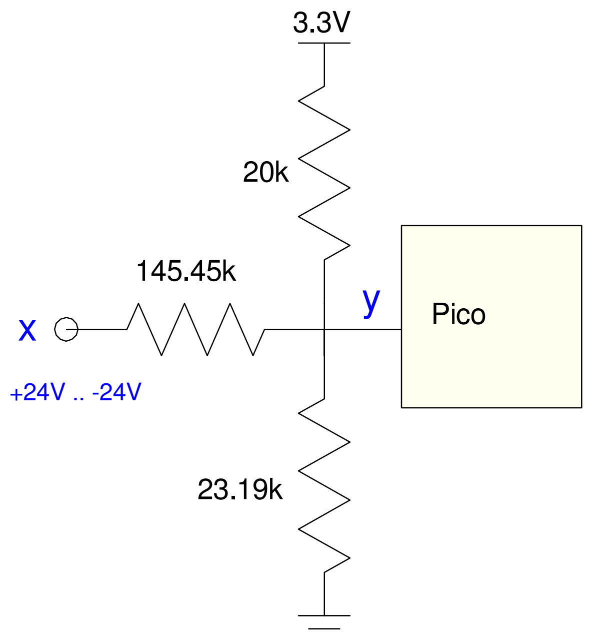
Assume a nominal resistance of 10k. Then

$$R_x = \left(\frac{10k}{0.06875}\right) = 145.5k$$

$$R_{3.3v} = \left(\frac{10k}{0.5}\right) = 20k$$

$$R_{0V} = \left(\frac{10k}{0.43125}\right) = 23.19k$$

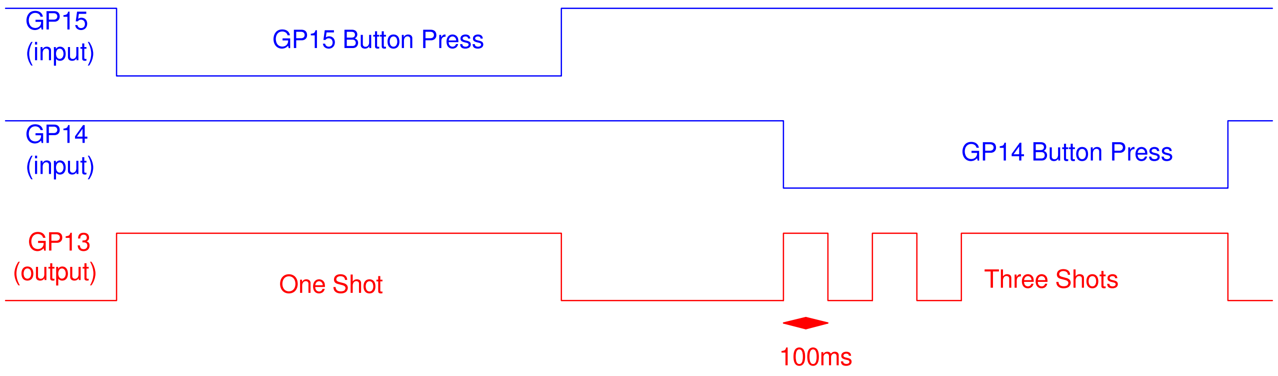
One solution is as follows (other solutions exist)



3) Fire Cheat: Write a Python program which controls the fire button for a video game

- When you press GP15, a single shot is output on GP13 (beeper)
- When you press GP14, three shots are output on GP13
  - On for 100ms, off for 100ms (twice), then on as long as GP14 is pressed
- Asume GP14 and GP15 are never pressed at the same time (don't care what happens in this case)

When the buttons are released, the beeper (GP13) turns off



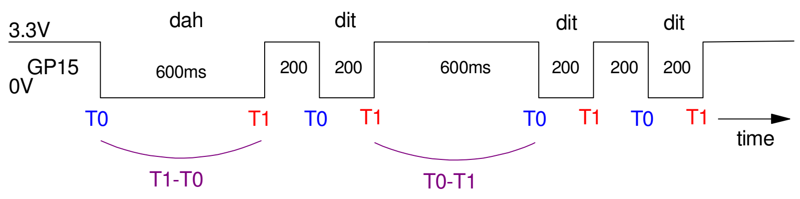
Code: (start with GP14 and 15 are inputs with a pull-down resistor, GP13 is output)

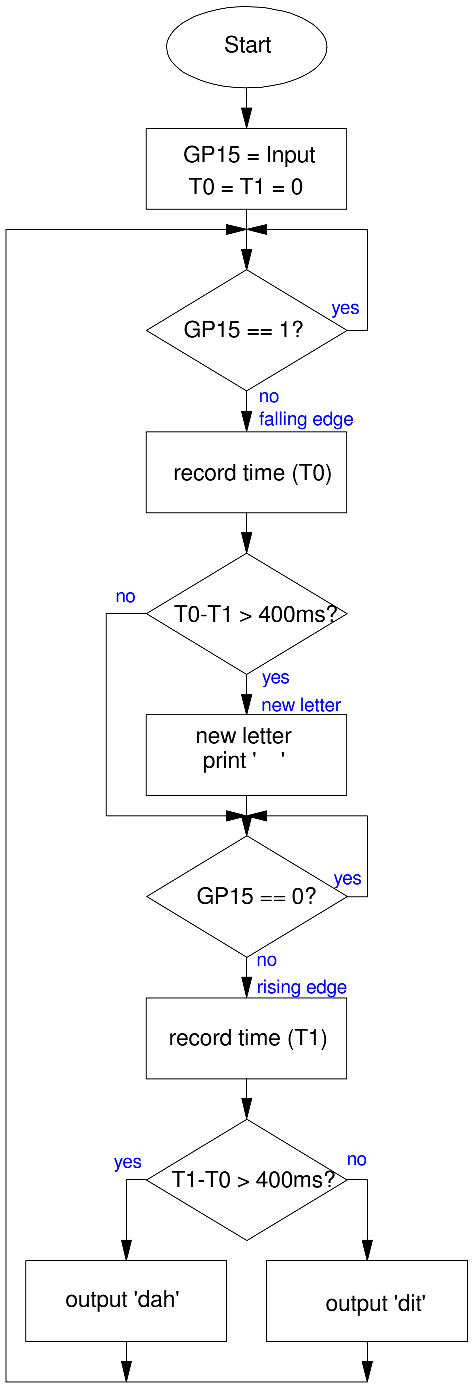
```
from machine import Pin
from time import sleep_ms

B15 = Pin(15, Pin.IN, Pin.PULL_UP)
B14 = Pin(14, Pin.IN, Pin.PULL_UP)
Beeper = Pin(13, Pin.OUT)

while(1):
    if(B15.value() == 0):
        Beeper.value(1)
        while(B15.value() == 0):
            pass
        Beeper.value(0)
    if(B14.value() == 0):
        for i in range(0,2):
            Beeper.value(1)
            sleep_ms(100)
            Beeper.value(0)
            sleep_ms(100)
        Beeper.value(1)
        while(B14.value() == 0):
            pass
        Beeper.value(0)
```

4) Morse Code: The flow chart for a program which outputs Morse code 'dit' and 'dah' based upon the duration of a signal is shown.

- 0V = button pressed (beeper on)
- 3.3V = button released (beeper off)

Write the corresponding Python program.





```
from machine import Pin
from time import time_ms

B15 = Pin(15, Pin.IN, Pin.PULL_UP)
T0 = T1 = 0

while(1):

     while(B15.value() == 1):
          pass
     T0 = time_ms()

     if(T0-T1 > 400):
          print('   ')

     while(B15.value() == 0):
          pass

     T1 = time_ms()

     if(T1 - T0 > 400):
          print('dah')
     else:
          print('dit')
```

Note: With bouncing, you get some stray dit outputs. To prevent that, change the last section of code to do nothing if T(on) < 10ms

```
     if(T1 - T0 > 400):
          print('dah')
     elif(T1 - T0 > 10):
          print('dit')
```

Generally Useful Python Routines

Binary Input (Button Pressed)

```
from machine import Pin

Button = Pin(15, Pin.IN, Pin.PULL_UP)
x = Button.value()
```

Binary Output (Blinking Light)

```
from machine import Pin

LED = Pin(16, Pin.OUT)
LED.toggle()
LED.value(1)
LED.value(0)
```

Analog Input (A2D Read)

```
from machine import ADC

a2d0 = ADC(0)
x = a2d0.real_u16()
```

Analog Output (PWM Output)

```
from machine import Pin, PWM

Aout = Pin(16, Pin.OUT)
Aout = PWM(Pin(16))
Aout.freq(1000)

# 0% duty cycle
Aout.duty_u16(0x0000)

# 100% duty cycle
Aout.duty_u16(0xFFFF)

# 50us pulse
Aout.duty_ns(50_000)
```

Measure a pulse width in micro-seconds

```
from machine import Pin, time_pulse_us

X = Pin(19, Pin.IN, Pin.PULL_UP)
low = time_pulse_us(19, 0, 500_000)
high = time_pulse_us(19, 1, 500_000)
```

Pause 1.23 seconds

```
from time import sleep

sleep(1.23)
```

For Loops

```
for i in range(0,6):
    d1 = i
    for j in range(0,4):
        d2 = j
        y = d1 + d2
```

While Loops

```
t = 0
while(t < 5):
    t = t + 0.01
    print(t)
```

If - else if - else statements

```
if(x < 10):
    a = 1
elif(x < 20):
    a = 2
else:
    a = 3
```

Random Numbers

```
from random import randrange

x = randrange(10)
# x = 0 to 9
```

Measure time since reset

```
from time import ticks_us

x0 = ticks_us()
```