Review for Test #2

ECE 476 Advanced Embedded Systems

Jake Glower

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

Format for Test #2

Five questions

- Edge Interrupt
- Timer Interrupt
- Analog Sensors - Hardware & Software
- Digital Sensor - Software
- LCD Graphics Display
- Random & Matrix Routines in Python

Available in-person or on BlackBoard

- In-Person
    - 50 minutes
    - Work problems in any order
    - Able to go back to probelms
- BlackBoard
    - 100 minutes
    - Random order with no backtracking
    - Must submit answers to first problem to move on to the next
    - Extra time due to no-backtracking, having to download, scan, upload problems

Edge Interrupts

- Rising Edge and/or Falling Edge
- Example: Up Counter

Define the pin to be input

Define the interrupt service routine
Usually need to pass data via global variables

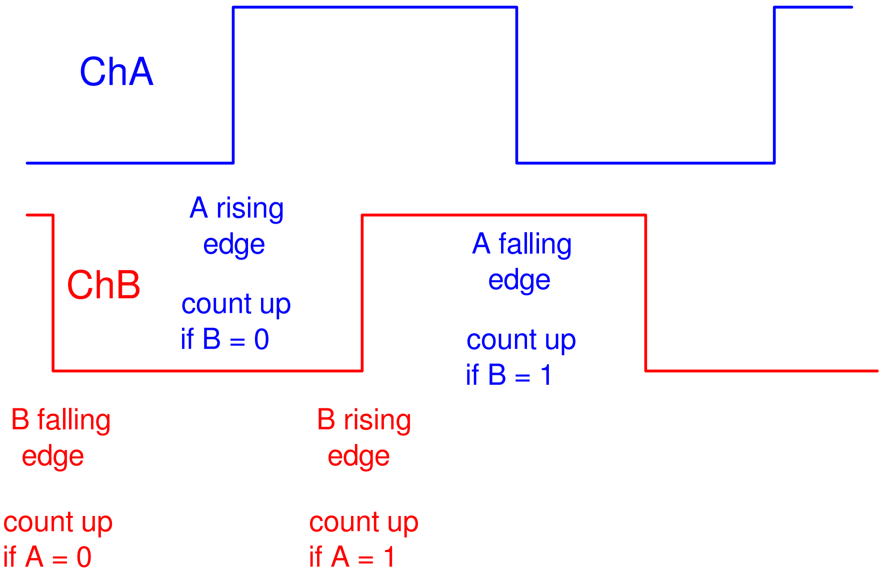
Set up the interrupt
IRQ_RISING
IRQ_FALLING

```
from machine import Pin

interrupt_flag=0
N = 0

pin = Pin(15,Pin.IN,Pin.PULL_UP)
def Count(pin):
    global interrupt_flag
    global N
    interrupt_flag=1
    N = N + 1

pin.irq(trigger=Pin.IRQ_FALLING,
handler=Count)

while(1):
    if(interrupt_flag):
        print("N = ", N)
        interrupt_flag=0
```

Edge Interrupts (cont'd)

Example: Optical Encoder

Edge interrupts can be for both rising and falling edges
You can have multiple edge interrupts turned on at the same time



```
N = 0

pin1 = Pin(15,Pin.IN,Pin.PULL_UP)
pin2 = Pin(14,Pin.IN,Pin.PULL_UP)

def ChA(pin1):
    global N
    if(pin1.value() == pin2.value()):
        N -= 1
    else:
        N += 1

def ChB(pin2):
    global N
    if(pin1.value() == pin2.value()):
        N += 1
    else:
        N -= 1

pin1.irq(trigger=Pin.IRQ_FALLING |
      Pin.IRQ_RISING, handler=ChA)
pin2.irq(trigger=Pin.IRQ_FALLING |
      Pin.IRQ_RISING, handler=ChB)
```

Timer Interrupts (periodic)

- Can trigger an interrupt every N seconds

Interrupt every 1.00 second

define a timer interrupt
Timer()

define the interrupt service routine
usually need global variables

Initialize the timer interrupt
interrupt rate (freq)
periodic interrupt
name of the int service routine

```
from machine import Pin, Timer
from time import sleep_ms

led = Pin(17, Pin.OUT)
tim = Timer()
N = 0

def tic(timer):
    global N
    N += 1

tim.init(freq=1, mode=Timer.PERIODIC,
callback=tic)

while(1):
    print(N)
    sleep_ms(100)
```

Timer Interrupts (one-shot)

- Can set up a single interrupt N second in the future
- Example: Turn off the buzzer 100ms in the future

declare a timer interrupt (tim)

declear inputs and outputs

define the interrupt service routine
turn off the buzzer

main loop:
wait for a button press

when detected turn on the buzzer

and set up a timer interrupt 100ms in the future

```
from machine import Pin, Timer

tim = Timer()

Buzzer = Pin(13, Pin.OUT)
Button = Pin(15, Pin.IN, Pin.PULL_UP)

def BuzzerOff(pin1):
    Buzzer.value(0)


while(1):
  while(Button.value() == 0):
    pass
  while(Button.value() == 1):
    pass

  Buzzer.value(1)

  tim.init(freq=10, mode=Timer.ONE_SHOT,
           callback=BuzzerOff)
```

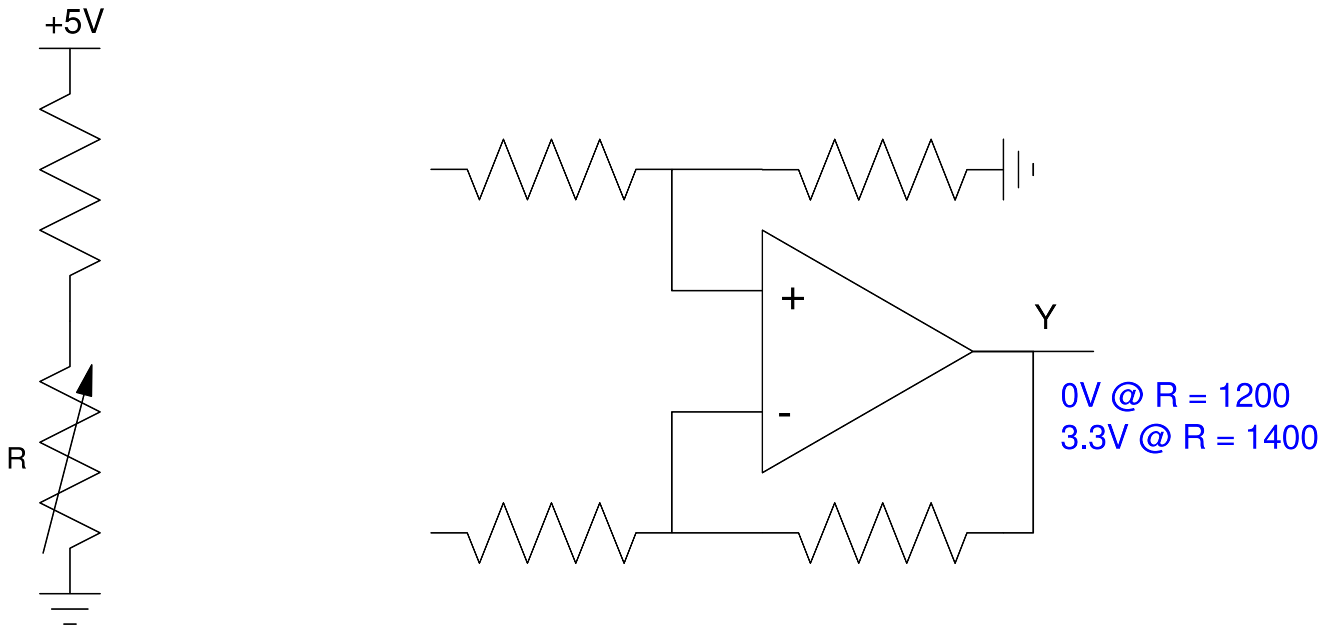
Analog Sensors (hardware)

Convert an analog signal to 0V to 3.3V range

- Range of the analog input on a Pi-Pico

Instrumentation Amplifier is commonly used

- note: circuit ground does not have to be earth ground



Analog Sensors (software)

Convert A/D reading to voltage

- 0x0000 = 0V
- 0xFFFF = 3.3V

Convert voltage to sensor units

- Ohms
- Lux
- Degrees C
- etc.

Example: Thermistor

$$R = 1000 \cdot \exp\left(\frac{3950}{T+273} - \frac{3950}{298}\right)\Omega$$

Example: TMP36

$$V = 0.5 + 0.01T$$

Digital Sensors

Many sensors have a digital interface

SPI & I2C:

- BME280 tempertature - pressure - humidity
- GY521 - accelerometer

For these sensors, you can

- Use bit-banging (manually set / clear bits)
- Use SPI and I2C functions in Python

You can often times find drivers online

Digital Sensors

Some digital sensors have non-standard interface:

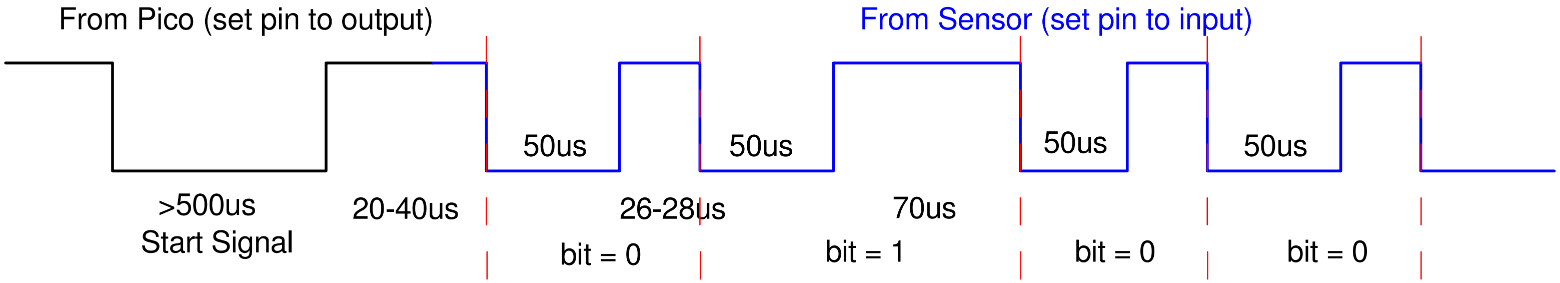
- HT11 and HT22

If you can't find a driver, you may need to write custom code to read the data

- Type of bit-banging

Example: HT11

- Logic 0: 26 - 28us pulse
- Logic 1: 70us pulse



LCD Graphics Display

- 480 x 320 display

Able to display text

```
LCD.Text('Hello World', 10, 50, Red, Black)
```

Able to draw lines

```
LCD.Line(5,5,200,200,Yellow)
```

Able to draw boxes

```
LCD.Box(1,1,479.319,White)
```

Random Library

MicroPython includes some random functions

- Additional random distributions can be created using these:

`randint(a,b)`	`returns an integer in the range of [a,b]`
`random()`	`returns a float in the range of (0,1)`
`randrange(a,b,dx)`	`returns a random number in the range of [a,b] with step size dx`
`uniform(a,b)`	`returns a float in the range of (a,b)`

Matrix Library

Python does not treat arrays as matricies

```
A = [0]
B = 5*A
print(B)
   B = [0,0,0,0,0]
```

You have to write your own routines to do matrix operations

- add, subrtract, multiply, inverse

These can be combined to do more complex matrix operations

- Least squares curve fitting
- Not as convenient as Matlab