# Electronics
# Interrupts and Threading

Terry Sturtevant

Wilfrid Laurier University

October 30, 2017

## Problems

## Problems

- If you use an ultrasonic distance sensor, what happens if there is no wall to detect?

## Problems

- If you use an ultrasonic distance sensor, what happens if there is no wall to detect?
- If you have a security system with break-in sensors, what does your code do most of the time?

## Problems

- If you use an ultrasonic distance sensor, what happens if there is no wall to detect?
- If you have a security system with break-in sensors, what does your code do most of the time?

**Useful Technique:** *Interrupts* and multiple *threads* allow you to create programs which don't waste lots of time waiting for unpredictable *events*.

## Problems

- If you use an ultrasonic distance sensor, what happens if there is no wall to detect?
- If you have a security system with break-in sensors, what does your code do most of the time?

**Useful Technique:** *Interrupts* and multiple *threads* allow you to create programs which don't waste lots of time waiting for unpredictable *events*. **Events** include specific transitions on GPIO pins.

- *Polling* is the process of checking in software for an *event*; i.e. a transition on an input.

- *Polling* is the process of checking in software for an *event*; i.e. a transition on an input.
- Polling precludes other operations while it is happening.

- *Polling* is the process of checking in software for an *event*; i.e. a transition on an input.
- Polling precludes other operations while it is happening.
- *Interrupts* allow the event to be detected by built-in hardware, *and redirects program control to code which deals with it automatically.*

- *Polling* is the process of checking in software for an *event*; i.e. a transition on an input.
- Polling precludes other operations while it is happening.
- *Interrupts* allow the event to be detected by built-in hardware, *and redirects program control to code which deals with it automatically.*

The main program and the *interrupt service routine* are called different *threads* of execution.

## Tips for interrupts

## Tips for interrupts

- Don’t have waiting in interrupt routines.

## Tips for interrupts

- Don’t have waiting in interrupt routines.
- Create flags to communicate between threads.

## Tips for interrupts

- Don’t have waiting in interrupt routines.
- Create flags to communicate between threads.
- Make interrupt routines as short as possible; have most processing done in the main thread.

## Example: Ultrasonic Sensor

## Example: Ultrasonic Sensor

Here's an example of using interrupts with the SR-04 ultrasonic distance sensor.

## Voltage levels

## Voltage levels

- The ultrasonic sensor is a 5V device.

## Voltage levels

- The ultrasonic sensor is a 5V device.
- A 3.3V trigger pulse *from* the Pi *may* work.

## Voltage levels

- The ultrasonic sensor is a 5V device.
- A 3.3V trigger pulse *from* the Pi *may* work.
- A 5V echo pulse *to* the Pi **is not OK!**

## Voltage levels

- The ultrasonic sensor is a 5V device.
- A 3.3V trigger pulse *from* the Pi *may* work.
- A 5V echo pulse *to* the Pi **is not OK!**

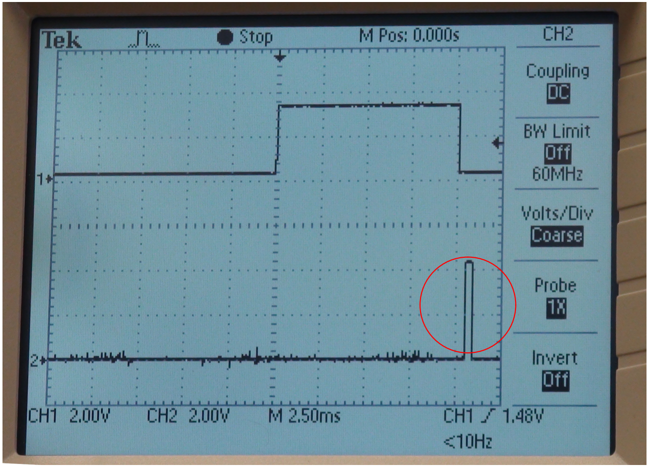
  **It could destroy the GPIO pin.**

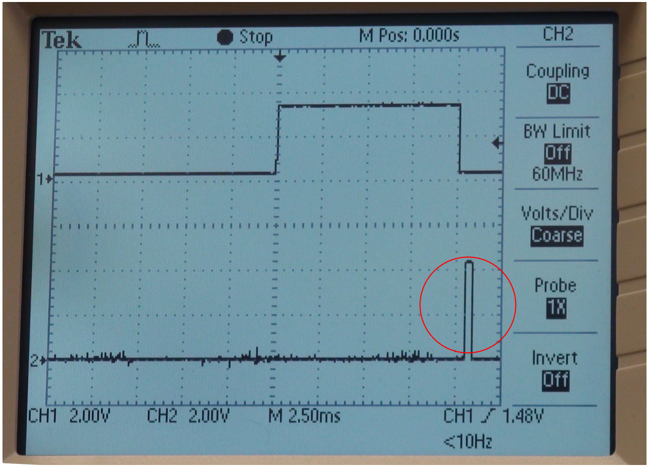
## Voltage levels

- The ultrasonic sensor is a 5V device.
- A 3.3V trigger pulse *from* the Pi *may* work.
- A 5V echo pulse *to* the Pi **is not OK!**

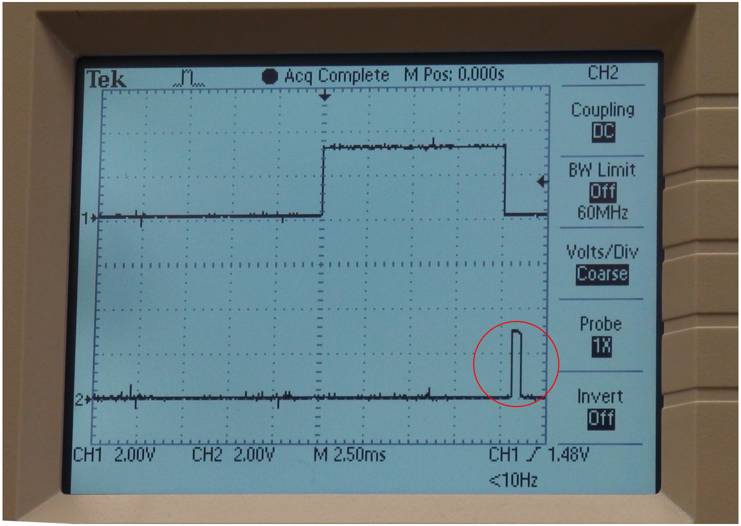
  **It could destroy the GPIO pin.**

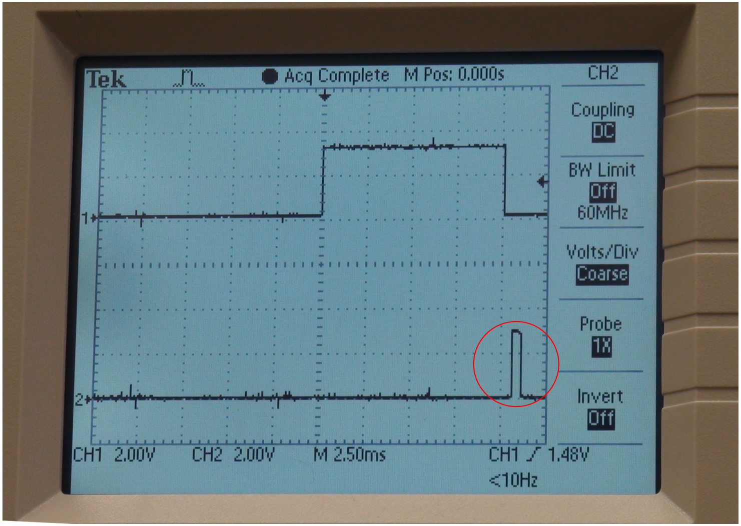
Optoisolating both signals takes care of both problems.


Tek
Stop
M Pos: 0.000s
CH2
Coupling
DC
BW Limit
Off
60MHz
Volts/Div
Coarse
Probe
1X
Invert
Off
CH1 2.00V
CH2 2.00V
M 2.50ms
CH1 ∕ 1.48V
<10Hz



Sensor side; 5V logic pulses


Tek
Acq Complete
M Pos: 0.000s
CH2
Coupling
DC
BW Limit
Off
60MHz
Volts/Div
Coarse
Probe
1X
Invert
Off
1
2
CH1 2.00V
CH2 2.00V
M 2.50ms
CH1 ╱ 1.48V
<10Hz



Raspberry Pi side; 3.3V pulses

## Interrupt sample code (1/4)

## Interrupt sample code (1/4)

```
 import RPi.GPIO as GPIO
 import datetime
 import time
#
#stuff omitted here
#
GPIO.setup(TRIGGER_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)
#
#stuff omitted here
#
```

## Interrupt sample code (2/4)

## Interrupt sample code (2/4)

```
#
#stuff omitted here
#
def Ultrasonic_Send_Pulse():
    GPIO.output(TRIGGER_PIN,GPIO.HIGH)
    time.sleep(PULSE_TIME)
    GPIO.output(TRIGGER_PIN,GPIO.LOW)
    global trigger_time
    trigger_time=datetime.datetime.now()
```

## Interrupt sample code (3/4)

## Interrupt sample code (3/4)

```
def Ultrasonic_Pulse_Received(channel):
    echo_time=datetime.datetime.now()
    delta=echo_time-trigger_time
    flight_time=delta.total_seconds()
    print "Rising edge detected on ECHO_PIN."
    print flight_time*speed
```

## Interrupt sample code (4/4)

## Interrupt sample code (4/4)

```
GPIO.add_event_detect(ECHO_PIN, GPIO.FALLING,
            callback=Ultrasonic_Pulse_Received)

try:
        while True:
            Ultrasonic_Send_Pulse()
            print "Waiting..."
            time.sleep(5)
            print "Done waiting..."

except KeyboardInterrupt:
        GPIO.cleanup()        # CTRL+C exit
```

## Interrupt Usage

## Interrupt Usage

- **GPIO.add_event_detect()**

  set up interrupt

## Interrupt Usage

- **GPIO.add_event_detect()**

  set up interrupt
- **ECHO_PIN**

  pin to monitor

## Interrupt Usage

- **GPIO.add_event_detect()**

  set up interrupt
- **ECHO_PIN**

  pin to monitor
- **GPIO.FALLING**

  transition (i.e. "event") to watch for

## Interrupt Usage

- **GPIO.add_event_detect()**

  set up interrupt
- **ECHO_PIN**

  pin to monitor
- **GPIO.FALLING**

  transition (i.e. “event”) to watch for
- **callback=Ultrasonic_Pulse_Received**

  function to execute when event happens

## Interrupt sample code

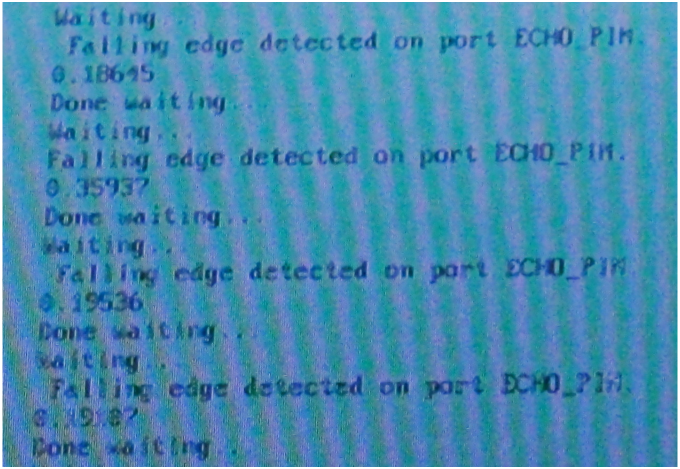
## Interrupt sample code

```
while True:
    Ultrasonic_Send_Pulse()
    print "Waiting..."
    time.sleep(5)
    print "Done waiting..."
```

## Interrupt sample code

```
while True:
    Ultrasonic_Send_Pulse()
    print "Waiting..."
    time.sleep(5)
    print "Done waiting..."
```

Note the *sleep* instruction after the pulse is sent.



```
Waiting...
 Falling edge detected on port ECHO_PIN.
0.18645
Done waiting...
Waiting...
Falling edge detected on port ECHO_PIN.
0.35937
Done waiting...
Waiting...
 Falling edge detected on port ECHO_PIN.
0.19536
Done waiting...
Waiting...
 Falling edge detected on port ECHO_PIN.
0.19187
Done waiting...
```

Transitions detected *during* sleep.

## Possible Improvements

## Possible Improvements

- There are many ways this code can be improved:

## Possible Improvements

- There are many ways this code can be improved:
  The time for the trigger pulse hasn't been optimized

## Possible Improvements

- There are many ways this code can be improved:
  The time for the trigger pulse hasn't been optimized
  The time between trigger pulses hasn't been optimized

## Possible Improvements

- There are many ways this code can be improved:

  The time for the trigger pulse hasn't been optimized

  The time between trigger pulses hasn't been optimized

No doubt there are many other possibilities as well.