Electronics
Pulse Width Modulation Sensors

Terry Sturtevant

Wilfrid Laurier University

February 1, 2018
Pulse Width Modulation Sensors

Analog information can be communicated over digital signals. This can be done by varying the width or spacing of digital pulses. This is called Pulse Width Modulation, PWM. This document gives a few examples.
Pulse Width Modulation Sensors

- Analog information can be communicated over digital signals
Pulse Width Modulation Sensors

- Analog information can be communicated over digital signals
  This can be done by varying the width or spacing of digital pulses
Pulse Width Modulation Sensors

- Analog information can be communicated over digital signals. This can be done by varying the width or spacing of digital pulses.
- This is called Pulse Width Modulation, PWM.
Pulse Width Modulation Sensors

- Analog information can be communicated over digital signals. This can be done by varying the width or spacing of digital pulses.
- This is called Pulse Width Modulation, PWM.
- This document gives a few examples.
Shaft encoders
Shaft encoders

- *Absolute* position sensing
Shaft encoders

- **Absolute** position sensing doesn’t use PWM
Shaft encoders

- *Absolute* position sensing doesn’t use PWM
- *Incremental rotary* encoding
Shaft encoders

- **Absolute** position sensing
doesn’t use PWM
- **Incremental rotary** encoding
uses PWM
Shaft encoders

- **Absolute** position sensing
  doesn’t use PWM
- **Incremental rotary** encoding
  uses PWM

As long as you know the initial position, you can update if you can sense changes.
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Terry Sturtevant
Electronics Pulse Width Modulation Sensors
Shaft encoder wheel
- Shaft encoder wheel
- Two sensors will allow determination of rotation *speed* and *angle*
Clockwise
Clockwise
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Clockwise
Clockwise
Counter-clockwise
Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Counter-clockwise
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Counter-clockwise
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

- Counter-clockwise
Shaft encoder timing

1

1
Shaft encoder timing

- 0
- 1
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

- Shaft encoder timing
  - 0
  - 1
- Shaft encoder timing
- 0
- 1
Shaft encoder timing

0

0
Shaft encoder timing

0

0
Shaft encoder timing

1

0
Shaft encoder timing

1

0
- Shaft encoder timing
- 1
- 0
Shaft encoder timing

1

1
- Speed of rotation from frequency of either channel
- *Speed* of rotation from frequency of either channel
- *Angle* of rotation from combination
- **Speed** of rotation from frequency of either channel
- **Angle** of rotation from combination

Here’s an example from an actual motor.
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Terry Sturtevant
Electronics Pulse Width Modulation Sensors
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Terry Sturtevant
Electronics Pulse Width Modulation Sensors
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Terry Sturtevant
Electronics Pulse Width Modulation Sensors
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters

Terry Sturtevant
Electronics Pulse Width Modulation Sensors
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control

Shaft encoders
Ultrasonic sensors
V to F and F to V converters
One direction
Other direction
Period is 5 divisions
Period is 5 divisions
Time scale is 500μS per division
Period is 5 divisions $\rightarrow 2500\mu S = 2.5mS$

Time scale is $500\mu S$ per division
Period is 5 divisions $\rightarrow 2500\mu S = 2.5mS$

30 slots per revolution $\times 2.5mS$
Period is 2.5mS

30 slots per revolution $\times$ 2.5mS $\rightarrow$ 75ms per revolution
Period is 2.5mS × 30 → 75ms per revolution
75/1000 seconds per revolution → 13.33 rev./second
Period is $2.5\text{mS} \times 30 \rightarrow 75\text{ms per revolution}$

$75/1000$ seconds per revolution $\rightarrow 13.33 \text{ rev./second}$

$\times 60 \rightarrow 800 \text{ RPM}$
Ultrasonic sensors
Ultrasonic sensors

- **Transmitter** sends out pulse
Ultrasonic sensors

- *Transmitter* sends out pulse
- *Receiver* registers echo
transmit
transmit
transmit
transmit
Pulse Width Modulation Sensors
Raspberry Pi Python PWM Control
Shaft encoders
Ultrasonic sensors
V to F and F to V converters

- transmit
- receive
transmit

receive
transmit

receive
- transmit
- receive
- *Time* from beginning of transmit to beginning of receive allows distance to be calculated
- Time from beginning of transmit to beginning of receive allows distance to be calculated.
- Since $2d = vt$
- Where $v$ is the speed of sound.
Time from beginning of transmit to beginning of receive allows distance to be calculated

since $2d = vt$

where $v$ is the speed of sound

Why is it $2d$?
V to F and F to V converters
V to F and F to V converters

- **V to F converters** take in an analog voltage and produce a string of digital pulses where the frequency is proportional to the input analog voltage.

- **F to V converters** take in a string of digital pulses and produce an analog voltage where the analog voltage is proportional to the input frequency.
V to F and F to V converters

- **V to F converters** take in an analog \textit{voltage} and produce a string of digital pulses where the \textit{frequency} is proportional to the input analog voltage.

- **F to V converters** take in a string of digital pulses and produce an analog \textit{voltage} where the analog voltage is proportional to the input \textit{frequency}.
Python PWM Control

```python
import GPIO

p = GPIO.PWM(channel, frequency)

p.start(dc)

p.ChangeFrequency(freq)

p.ChangeDutyCycle(dc)

p.stop()
```
Python PWM Control

- \( p = \text{GPIO.PWM}(\text{channel}, \text{frequency}) \)
  
  open channel at given frequency
Python PWM Control

- \[ p = \text{GPIO.PWM}(\text{channel, frequency}) \]
  open channel at given frequency

- \[ p.\text{start}(dc) \]
  start at given duty cycle (\textit{percent})
Python PWM Control

- \( p = \text{GPIO.PWM}(\text{channel, frequency}) \)
  open channel at given frequency
- \( p.\text{start}(dc) \)
  start at given duty cycle (\text{percent})
- \( p.\text{ChangeFrequency}(freq) \)
  change frequency
Python PWM Control

- \[ p = \text{GPIO.PWM}(\text{channel}, \text{frequency}) \]
  open channel at given frequency
- \[ p.\text{start}(dc) \]
  start at given duty cycle (percent)
- \[ p.\text{ChangeFrequency}(\text{freq}) \]
  change frequency
- \[ p.\text{ChangeDutyCycle}(dc) \]
  change duty cycle (percent)
Python PWM Control

- \( p = \text{GPIO.PWM}(\text{channel, frequency}) \)
  open channel at given frequency
- \( p.\text{start}(dc) \)
  start at given duty cycle (percent)
- \( p.\text{ChangeFrequency}(freq) \)
  change frequency
- \( p.\text{ChangeDutyCycle}(dc) \)
  change duty cycle (percent)
- \( p.\text{stop}() \)
  stop PWM
Python PWM sample code

```python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BORDER)
GPIO.setup(12, GPIO.OUT)
p = GPIO.PWM(12, 50)  # chan=12 freq=50Hz
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```
Python PWM sample code

```python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)
p = GPIO.PWM(12, 50)  # chan=12 freq=50Hz
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```
Raspberry Pi PWM pins
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
- **PWM0** uses (BOARD) 12 and 32.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
- **PWM0** uses (BOARD) 12 and 32.
  i.e. (BCM) GPIO18 and (BCM) GPIO12.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
- **PWM0** uses (BOARD) 12 and 32.
  i.e. (BCM) GPIO18 and (BCM) GPIO12.
- **PWM1** uses (BOARD) 33 and 35.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
- **PWM0** uses (BOARD) 12 and 32.
  i.e. (BCM) GPIO18 and (BCM) GPIO12.
- **PWM1** uses (BOARD) 33 and 35.
  i.e. (BCM) GPIO13 and (BCM) GPIO19.

Note: These pins are shared with the audio subsystem.
Raspberry Pi PWM pins

- The Raspberry Pi has 2 built-in PWM channels.
- Each channel has 2 associated pins.
  (So only one pin can be used at a time.)
- **PWM0** uses (BOARD) 12 and 32.
  i.e. (BCM) GPIO18 and (BCM) GPIO12.
- **PWM1** uses (BOARD) 33 and 35.
  i.e. (BCM) GPIO13 and (BCM) GPIO19.

Note: These pins are shared with the audio subsystem.