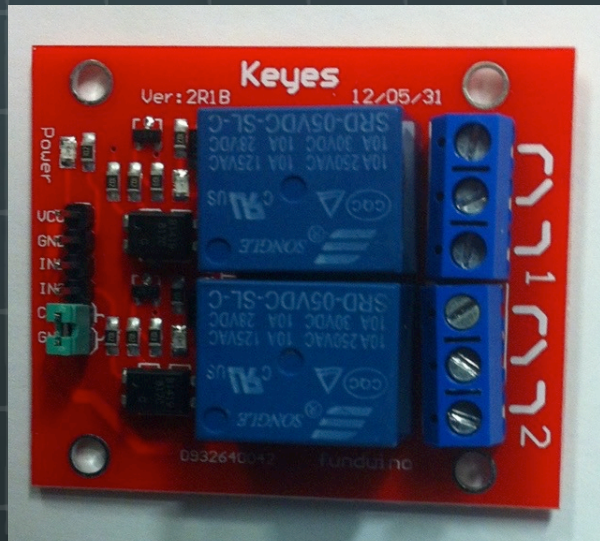
# C++ Code Snippets PART II: Outputs for Arduino IDE/Teensy 3.2

John R. Wright, Jr., PhD, CSTM, CLSSGB, CSCE, F.ATMAE
AENG 467, Mobile Robotics

# Relay

## (Controlling an external output like the Fan)

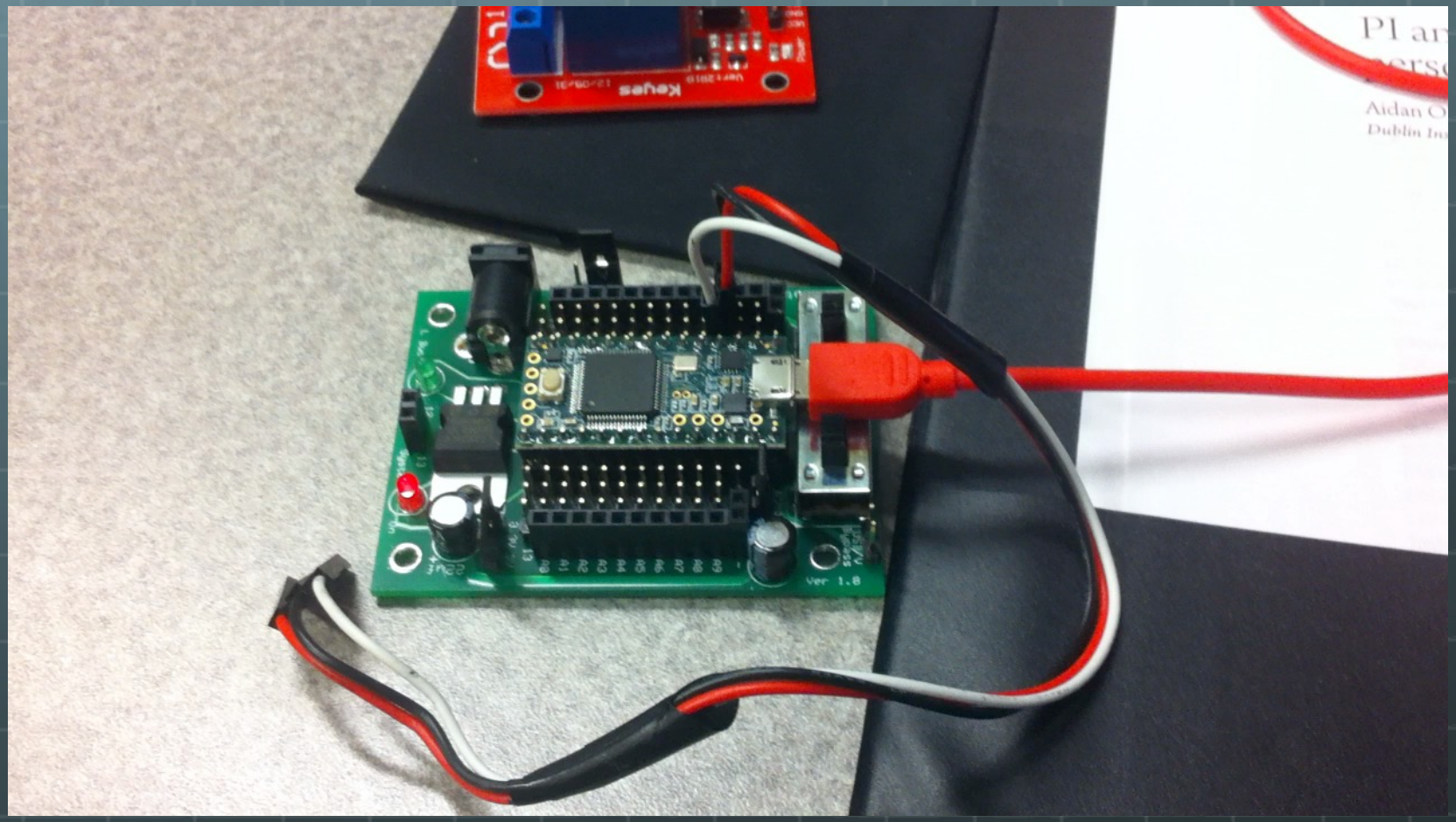
```
// John Wright 2017
// January 19, 2017
// Controlling/cycling a Relay on and off

int Relay = 2;                  //Where device is connected on-board

void setup() {
  pinMode(Relay, OUTPUT);       //Set the pin direction to output
}

void loop() {
  digitalWrite (2, HIGH);       //Click relay coil on
  delay (1000);                 //1 sec
  digitalWrite (2, LOW);        //Click relay coil off
  delay (2000);
}
```



https://youtu.be/y3j6EuRbGCY

# Relay

## (Controlling an external output like the Fan)



# Servo Motor

Reference: Ch 18, Chris Odom's Vol 2 Book

"As you learned earlier in this chapter, most servos have a minimum pulse width limit around 1.0ms and a maximum limit around 2.0ms, although the actual minimum and maximum pulse widths may vary between the various servo brands. Therefore, when programming servos it is important to keep the range of the values sent to the analogWrite() function between 127 and 255 , which corresponds to a pulse width of 1000µs and 2000µs , respectively!"

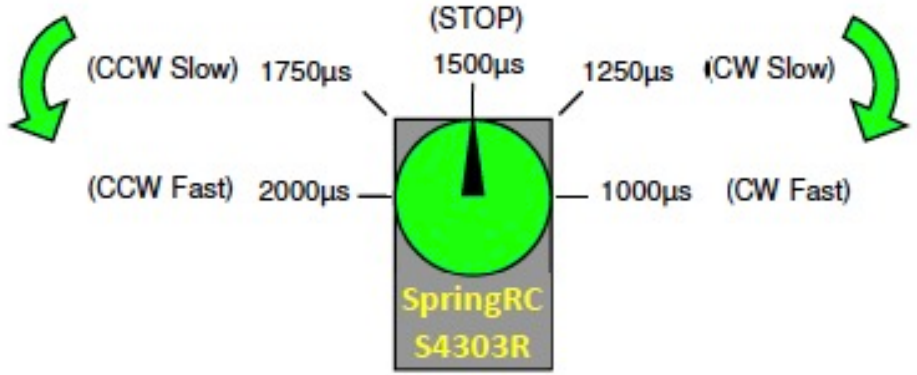



**Figure xxx.** The relationship between pulse widths and the rotational speed and direction of the SW-S4303R servo from SpringRC.

# Servo Motor via analogWrite()

```
//Chris Odom, 2016

byte servoPin = 3;                    // For this sketch, the servo MUST be a in a PWM pin!

void setup() {
  pinMode(servoPin, OUTPUT);
}

void loop() {
  analogWrite(servoPin, 255);      // Servo will spin CCW fast
}
```

# Servo Motor via digitalWrite()

```
/*
Chris Odom, 2016 Ch 18 Vol2
Edited and expanded by John Wright, 2017
because this module uses digitalWrite (not analogWrite) to
control the servos, you can attach the servos to ANY pins (not just PWM pins)!
*/

const int leftServo = 0;
const int rightServo = 1;

// servo direction constants (1000-2000 where 1500 is stop)
const int left_forward_fast = 2000;          // CCW
const int left_backward_fast = 1000;         // CW
const int right_forward_fast = 1000;         // CW
const int right_backward_fast = 2000;        // CCW
int x = 0;
int y = 0;
```

# Servo Motor via digitalWrite()

```
void setup() {
  pinMode(leftServo,OUTPUT);
  pinMode(rightServo,OUTPUT);
}
// This is your Main Program that is calling subroutines
void loop() {
  forwardStepFast();
  delay(100);
  backwardStepFast();
  delay(100);
}
```

# Servo Motor via digitalWrite()

From Chris Odems's Ch 18 Vol 2 text

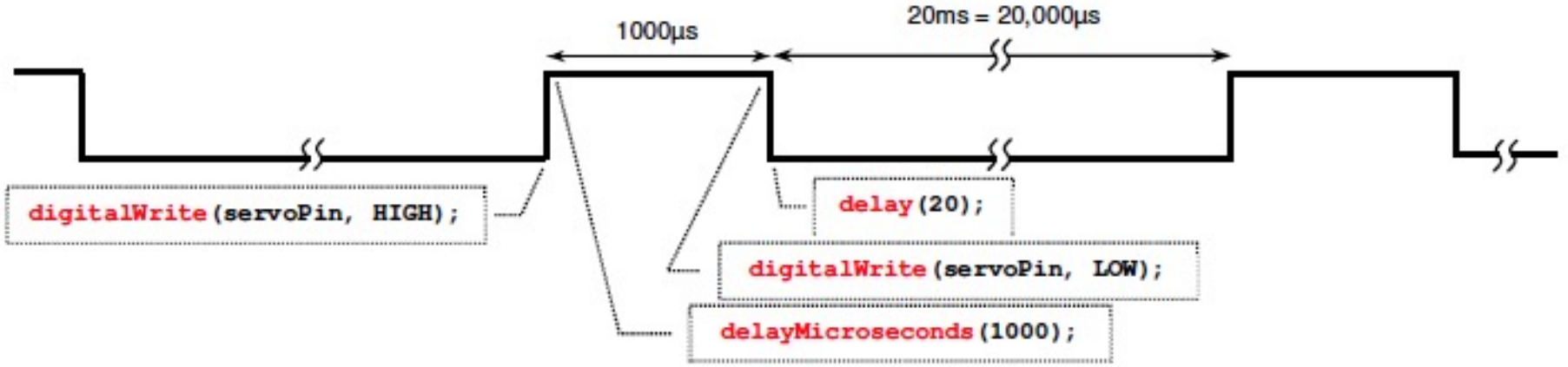



**Figure xxx.** A graphical representation of a train of 1000µs-pulses generated with `digitalWrite()` and `delay()` commands within a loop. A necessary delay of 20ms (20,000µs) separates each pulse to give time for the servo to rotate. Each pulse turns the servo's shaft rapidly clockwise.

# Servo Motor via digitalWrite()

```
//*************************************************************
// This is a subroutine for forward
//For-Loop below - see page 285 Vol 1

void forwardStepFast() {
for (int x = 0 ; x < 100 ; x++) {
      servoMove(leftServo, left_forward_fast); //Calling another function/subroutine
      servoMove(rightServo, right_forward_fast);
      delay(20); // This value changes speed of motor, do not set < 20ms
  }
}
```

# Servo Motor via digitalWrite()

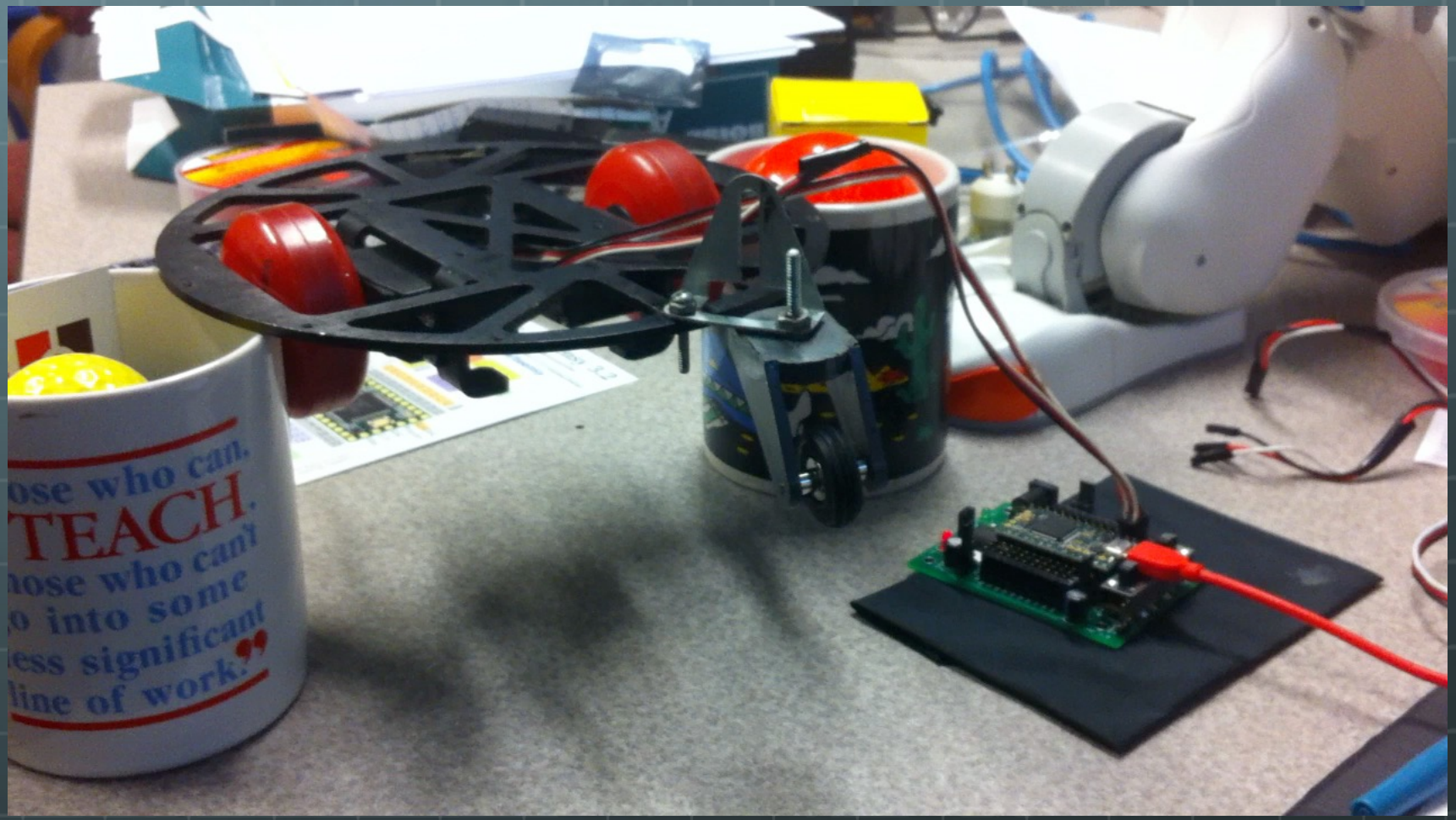
```
//**************************************************
// This is a subroutine for backwards

void backwardStepFast() {
for (int y = 0 ; y < 100 ; y++) {
    servoMove(leftServo, left_backward_fast);
    servoMove(rightServo, right_backward_fast);
    delay(20); // This value changes speed of motor, do not set < 20ms
  }
}
```

# Servo Motor via digitalWrite()

```
//*************************************************
// subroutine that defines one step with one servo

void servoMove(byte servoPin, int pulseWidth) {
  digitalWrite(servoPin, HIGH); // create the rising edge of the pulse
  delayMicroseconds(pulseWidth); // set pulse width in microsec
  digitalWrite(servoPin, LOW); // create the falling edge of the pulse
}
```

# Servo Motor via digitalWrite()

https://youtu.be/0UQ0a6u3cKg



# When to use analogWrite() and when to use digitalWrite() to Control Servos

by Chris Odom

Physical Computing & Robotics with the Arduino IDE Vol 2

- "Using analogWrite() to spin a servomotor is sometimes the perfect function to use. This is true, for example, when
  - perpetual motion is called for or when the motion is time-based. Here, one line of code will cause the servo to spin forever.
  - This is quite handy when you need a siren, or flashing lights, or merry-go-rounds, or floor scrubbers – something that you want
  - to start and then forget about."

# When to use analogWrite() and when to use digitalWrite() to Control Servos

by Chris Odom

Physical Computing & Robotics with the Arduino IDE Vol 2

- "However, analogWrite() is not well-suited for step-based events, such as blinking the LED five times. In my experience, driving the wheels of a robot lends itself to a more step-based operation. For example, when your robot is traversing a tabletop it should scan for the table's edge with every step, rather than some arbitrary time interval!

- Another reason I'm not fond of using analogWrite() to drive a servo is the lack of resolution and precision. In the above function, changeSpeedsAnalogW(), I showed how a wide range of values yielded identical servo speeds. Servos are not terribly precise devices in the best of circumstances, but using analogWrite() for high-precision motion is not advisable.

- Another of the main drawbacks of using analogWrite() to program servos is the fact that the servo must be connected to one the PWM pins on your development board.8 Often when working on large projects, the microcontroller's signal pins get consumed by a wide array of sensors, actuators, and motors and finding a free PWM pin can be problematic."

# PWM via a Library

Servo_Library_Test_Code_Sumobot_6_2_2021

```
/*
Servo Library Test Code - Sumobot
Dr. John Wright
6/2/2021
*/

#include <Servo.h>
Servo leftservo;                        // create servo object to control our left servo
int spd = 110;                          // variable to store the servo speed 0 = full reverse, 180 is full forward, ~90 is stop

void setup() {
    leftservo.attach(2);                // leftservo connected to pin 2
}

void loop() {

    leftservo.write(spd);               // tell servo to go to position in variable 'pos'
    delay(20);                          // wait 20ms for the servo off-time to protect servo

}
```

# Good Luck!
# This is Engineering!

https://www.youtube.com/watch?v=nFbWXuR_2Ow