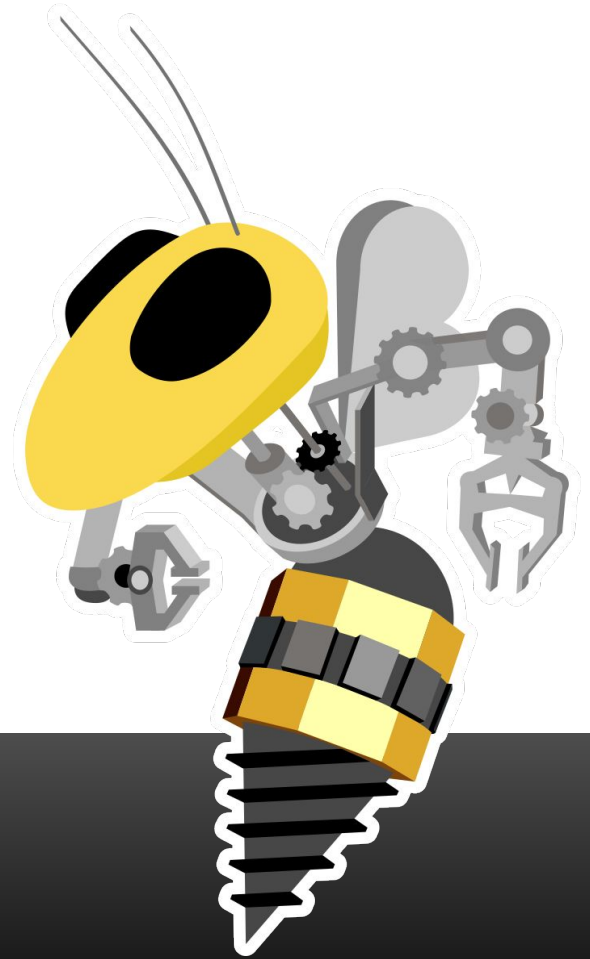


# Welcome!

Firmware Training  
Week 3

**ROBOJACKETS**  
COMPETITIVE ROBOTICS AT GEORGIA TECH

*[www.robojackets.org](http://www.robojackets.org)*



# Last Week!

- C++ Continued
- Interrupts
- State Machines

# This Week!

- Registers and Timers
- PWM
- Binary and Hexadecimal
- Bitwise Operations
- Masking
- Lab

# Dues

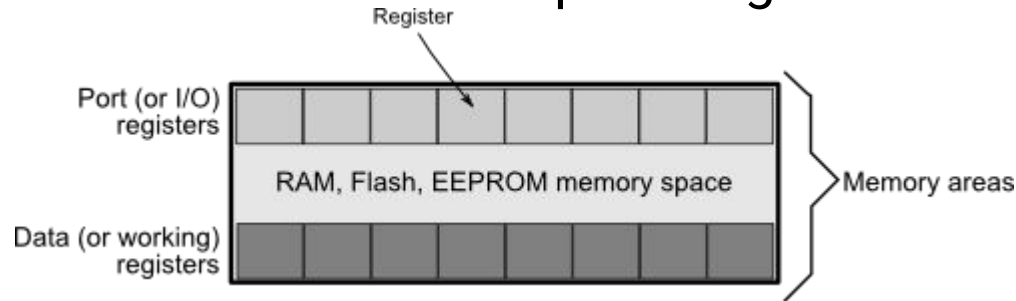
- Please pay dues by Saturday, October 10th
- Can do so at [my.robojackets.org/dues](https://my.robojackets.org/dues)



# Registers and Timers

# What is a register?

- Hardware used to read and write multiple bits at once
- Used to form memory to store information
- Microcontrollers often use registers to control the functions of various parts of the device
- Methods like Arduino's `digitalWrite` modify the value of a register that controls whether a pin is high or low



# Timers and Timer Registers

- Timers are a peripheral device used to count up to a certain value before resetting
- Timer registers are registers that act as a timer
- Timer registers can be used to trigger different actions at set intervals by checking the value of the timer register against a different register



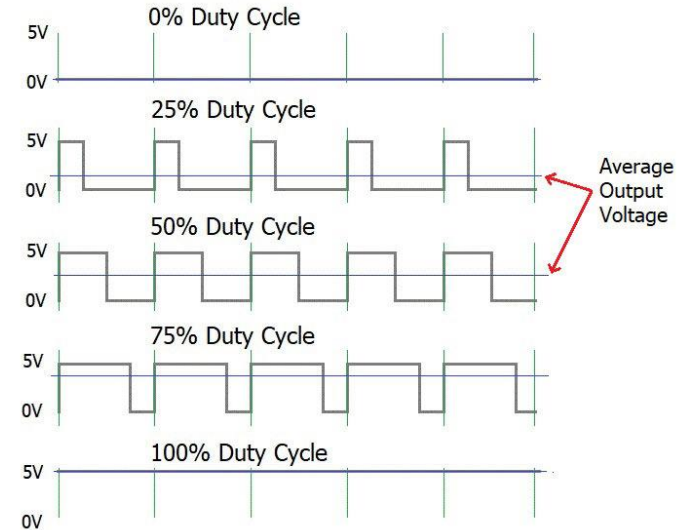
# PWM

Pulses go brrrrr



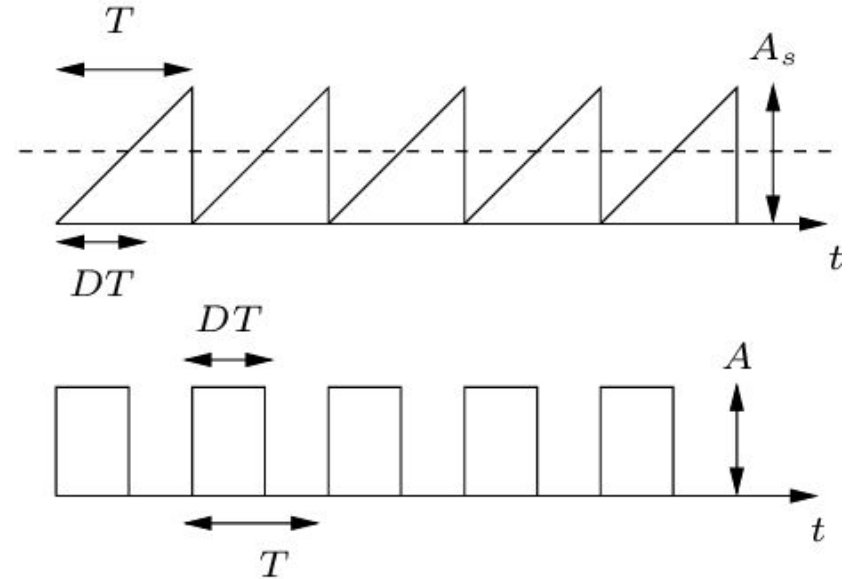
# What is PWM?

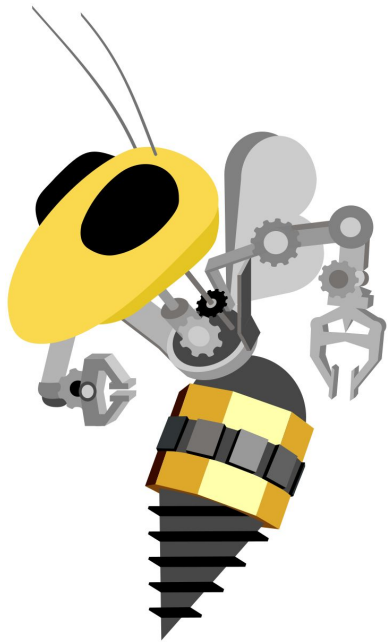
- Pulse-width modulation
- By alternating between a low and high voltage the effect of a middle voltage can be achieved
- By varying the duty cycle of this alternation the spectrum of voltage between the low and high can be utilized. (The average value of the voltage)



# Relation to Timers

- Switching frequency is controlled by a timer register
- The time voltage is high compared to the total time tells us the % duty cycle which directly relates to the output voltage





# Binary and Hexadecimal

52 6f 62 6f 4a 61 63 6b 65 74 73

# What are Binary and Hexadecimal?

- Binary and Hexadecimal are alternate number systems
- The number system that everyone is taught is called decimal and it is base 10
- Binary is base 2 : 0,1
- Hexadecimal is base 16 : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

decimal	hexadecimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

# Example Conversion Binary-Decimal

Decimal to Binary: 158 to  
10011110

$$158 / 2 = 79 \quad | r = 0$$

$$79 / 2 = 39 \quad | r = 1$$

$$39 / 2 = 19 \quad | r = 1$$

$$19 / 2 = 9 \quad | r = 1$$

$$9 / 2 = 4 \quad | r = 1$$

$$4 / 2 = 2 \quad | r = 0$$

$$2 / 2 = 1 \quad | r = 0$$

$$1 / 2 = 0 \quad | r = 1$$



Binary to Decimal: 10011110 to  
158


$2^7$ 128	$2^6$ 64	$2^5$ 32	$2^4$ 16	$2^3$ 8	$2^2$ 4	$2^1$ 2	$2^0$ 1
1	0	0	1	1	1	1	0

$$128*1 + 16*1 + 8*1 + 4*1 + 2*1 = 158$$

# Example Conversion Hexa-Decimal

Decimal to Hexadecimal:

158 to 9E

$$\begin{array}{l} 158 / 16 = 9 \quad | \quad r = 14 \\ 9 / 16 = 0 \quad | \quad r = 9 \end{array}$$


14 is E in Hexadecimal so  
the result is 9E !

Hexadecimal to Decimal: 9E  
to 158

$16^1$ 16	$16^0$ 1
9	E

$$9 * 16 + 14 * 1 = 144 + 14 = 158$$

# Example Conversion Hexa-Binary

Hexa to Binary :  
9E to 10011110

9  
↓  
1001

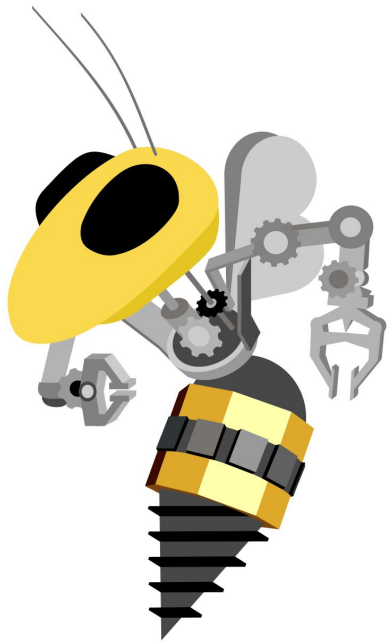
E  
↓  
1110

decimal	hexadecimal	binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Binary to Hexa :  
10011110 to 9E

1001  
↓  
9

1110  
↓  
E



# Bitwise Operations

Yes or no



# What are Bitwise Operations?

- Bitwise operators:
  - AND  $\rightarrow$  &
  - OR  $\rightarrow$  |
  - XOR  $\rightarrow$  ^

BIT A	BIT B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Bitwise operations are Logical Operations but instead of comparing conditions, bits are compared

# How does it work?

Example: 01000011(67) ; 00101010(42)

01000011	01000011	01000011
$\&$ 00101010	00101010	$\wedge$ 00101010
<u>00000010</u>	<u>00101010</u>	<u>00101010</u>
00000010	01101011	01101001

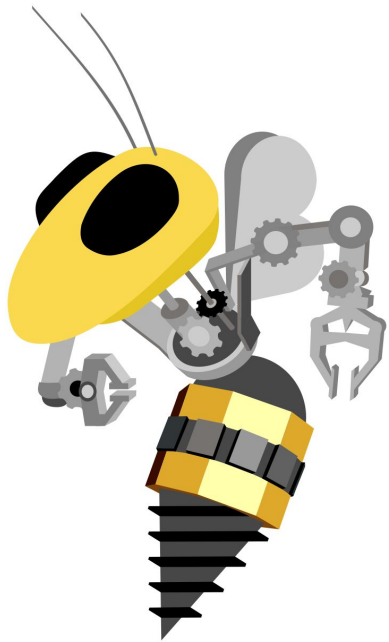
# Logical Shift

Left Shift	Right Shift
<<	>>>

- Left Shift
  - A single left shift multiplies a number by 2:
    - (2) 0010 << 1 → 0100 (4)
- Right Shift
  - A single right shift divides a number by 2 (discards the remainder)
    - (5) 0101 >>> 1 → 0010 (2)

# Why are they useful?

- Very efficient
- Small memory usage
- They are used in embedded devices, socket programming (network), cryptography etc...



# Masking

Mask on (or off)

# What is it?

- Using bitwise operations to access/set particular bits
- Read a value from a binary number
  - Masking with 0 to ignore subset of bits
  - Use a bitwise AND (&)
- Write certain bits of a binary number
  - Masking bits to 1 to change subset of bits
  - Use a bitwise OR (|)

# How to mask?

Masking with &:

- Any number & 0 leaves a 0
- Any number & 1 leaves that number

Ex:

```
1001010
&0101001
0001000
```

Masking with |:

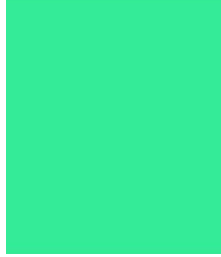
- Any number | 0 leaves that number
- Any number | 1 leaves a 1

Ex:

```
1001010
| 0101001
1101011
```

# Example

You have a color : #34EB98 (hexadecimal),  
1101001110101110011000 (binary)



**How do you get only the amount of blue of this color ?**



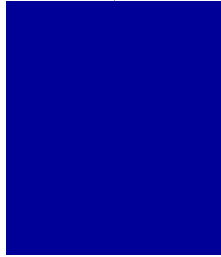
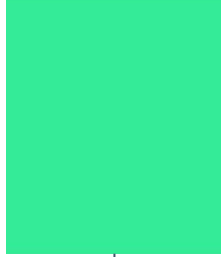
# Example

You have a color : #34EB98 (hexadecimal),

001101001110101110011000 (binary)

**How do you get only the amount of blue of this color ?**

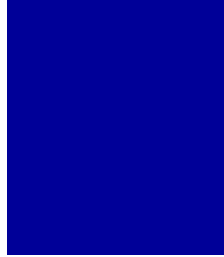
Mask (&) :      0011 0100 1110 1011 1001 1000  
                 & 0000 0000 0000 0000 1111 1111  
                 => 0000 0000 0000 0000 1001 1000



# Example

Now we have :      #000098 (hexadecimal),  
000000000000000010011000 (binary)

**How do you add red to this color ?**

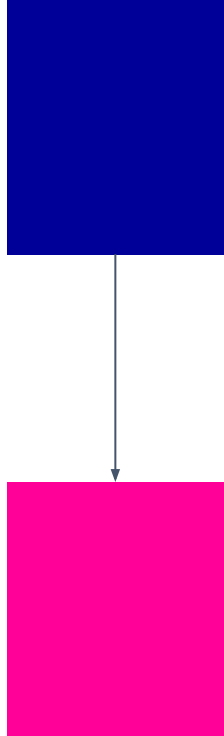


# Example

Now we have :      #000098 (hexadecimal),  
000000000000000010011000 (binary)

**How do you add red to this color ?**

Mask (I) :      0011 0100 1110 1011 1001 1000  
                 |    1111 1111 0000 0000 1111 1111  
=>      1111 1111 0000 0000 1001 1000





# Lab Time

# Lab Info

- Read about ATmega microcontroller
- Read about the timer registers
  - Setup fast PWM mode using the register
- Write interrupt handler for button
- Display brightness using LED