Programming BASIC Stamp I

SS-3406 Introduction to Robotics

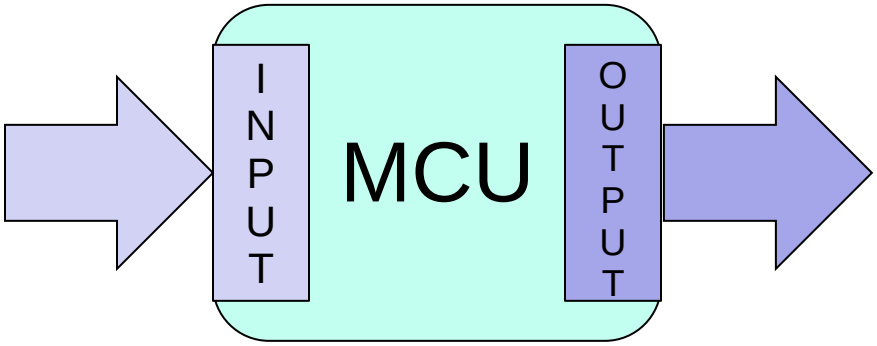
Programming?!

MCU: BASIC STAMP 2

Robot Brain: Microcontroller

- **Microcontroller** or Microcontroller Unit (**MCU**) is the electronic device that act as the **brain** of a robot.
- MCU is a computer on a single integrated circuit (IC) chip.
- It reads sensors as its **inputs** and controls its actuators as its **outputs**.

Inputs: sensors' data, e.g. switches, are fed in through the **input ports/pins**.



Outputs: signals to activate the actuators, e.g. turn on a light, are sent out through the **output ports/pins**.

Robot Program

- Remember **MCU**s are computer?
  - So, a robot program is a computer program
- **Computer program** is a **sequence of instructions** for a computer (MCU) to follow.
  - Note instructions lead to events.
  - E.g.
    Read the bumper sensors,
    if they indicate contact,
    stop the wheels.

```
task main()
{
  bMotorReflected[port2]=1;
  while(true)
  {
    if(SensorValue(bumper)==0)
    {
      motor[port3]=127;
      motor[port2]=127;
    }

    else
    {
      motor[port3]=127;
      motor[port2]=-127;
      wait1Msec(1500);
    }
  }
}
```

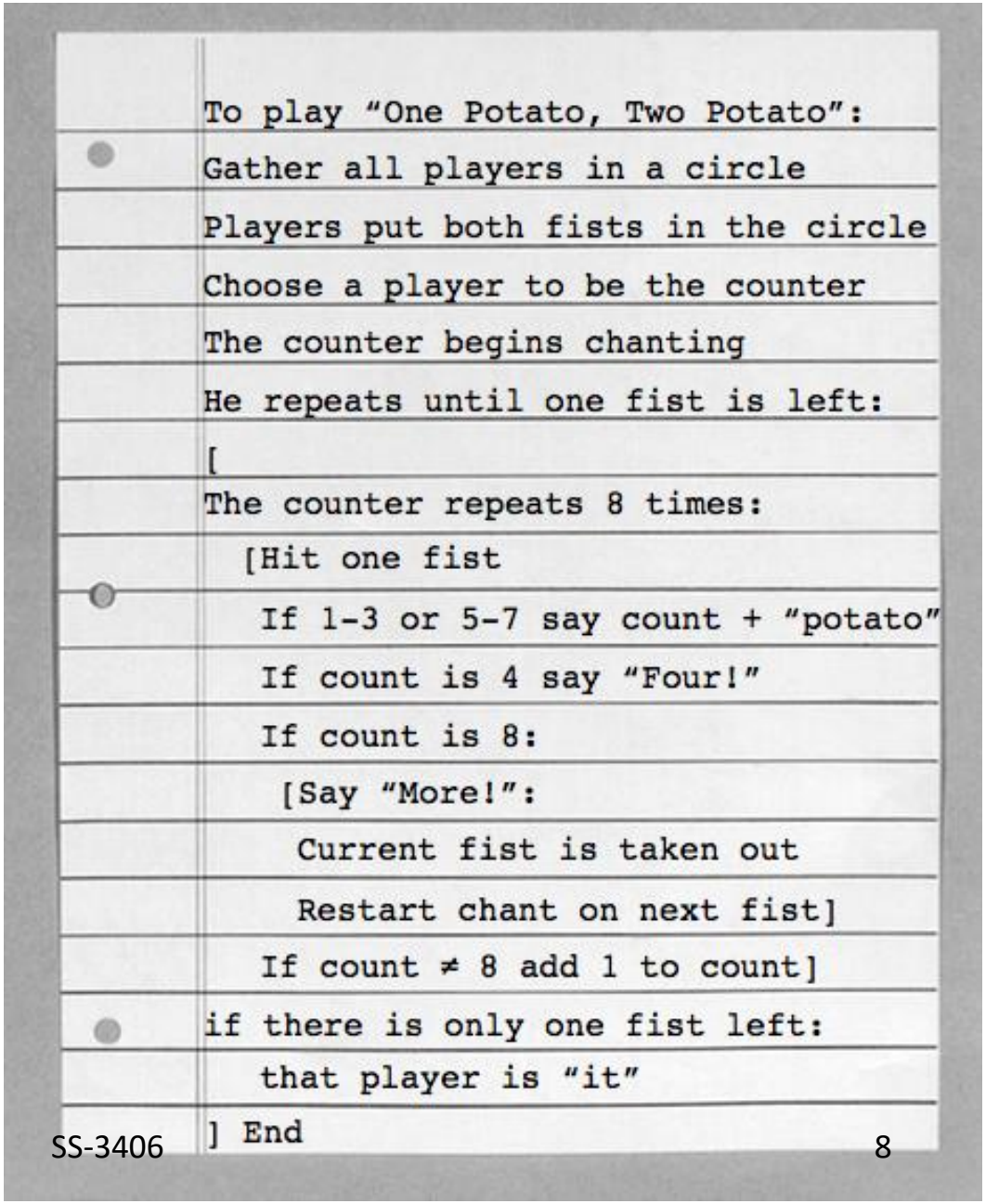
Programming a robot is about reading the input ports (sensors), understand the inputs (perception), and deciding (control) what actions to be taken at the output ports (actuators).

Programming in Five Steps

1. What? What exactly do you want to program?
   - E.g. Robot to follow a white line.
2. How? Design the program.
   - Determine program logic (flow).
   - Design details using flowchart and/or pseudocode.
3. Write it. Code the program.
   - Know the language, know the IDE.
4. Test the program. Debugging.
5. Document and maintain.

Pseudocode

- Normal language (our language) statements to describe the program logic, i.e. the flow of the sequence of events or instructions.
- Translates our thinking to program.



```
To play "One Potato, Two Potato":
Gather all players in a circle
Players put both fists in the circle
Choose a player to be the counter
The counter begins chanting
He repeats until one fist is left:
[
The counter repeats 8 times:
  [Hit one fist
   If 1-3 or 5-7 say count + "potato"
   If count is 4 say "Four!"
   If count is 8:
    [Say "More!":
     Current fist is taken out
     Restart chant on next fist]
   If count ≠ 8 add 1 to count]
if there is only one fist left:
   that player is "it"
] End
```

When you write a program, think of it as you are teaching a kid to perform a task – giving them every steps.

That's the pseudocode to your program.

Programming Concepts

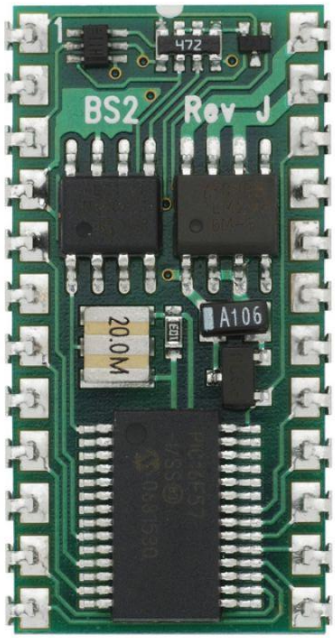
- **Sequence** of instructions
  - Get the order correct.
- Program **flow control**
  - **Conditional structure**: do certain things based on a true or false, yes or no decision.
  - **Looping structure**: a list of instructions to do more than once.
- **Program structure** – language specific
  - The template: different sections of the source code. Block of codes.
- Program **syntax** – language specific
  - Instructions, symbols and statements in the source code.
- How to deal with the **data**: **variables**, **constant**, **data structures**.(e.g. array).

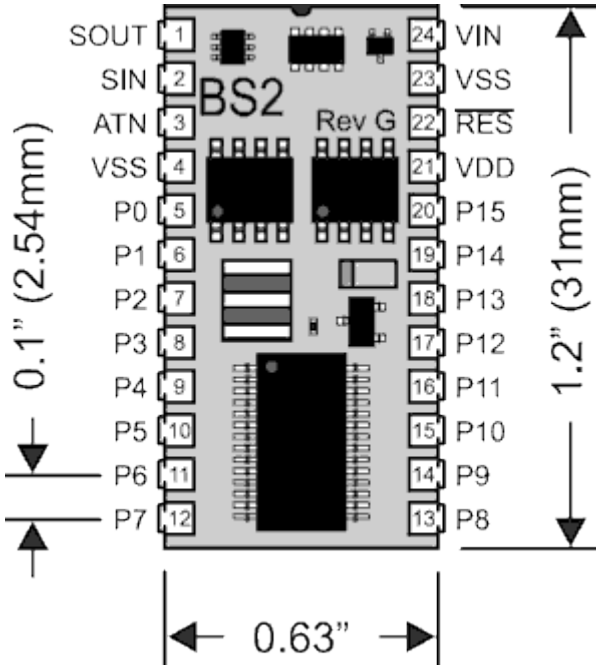
Programming a Microcontroller

- Three things required:
  - Microcontroller (MCU)
  - Programmer (Hardware)
  - Programming Environment (Software)

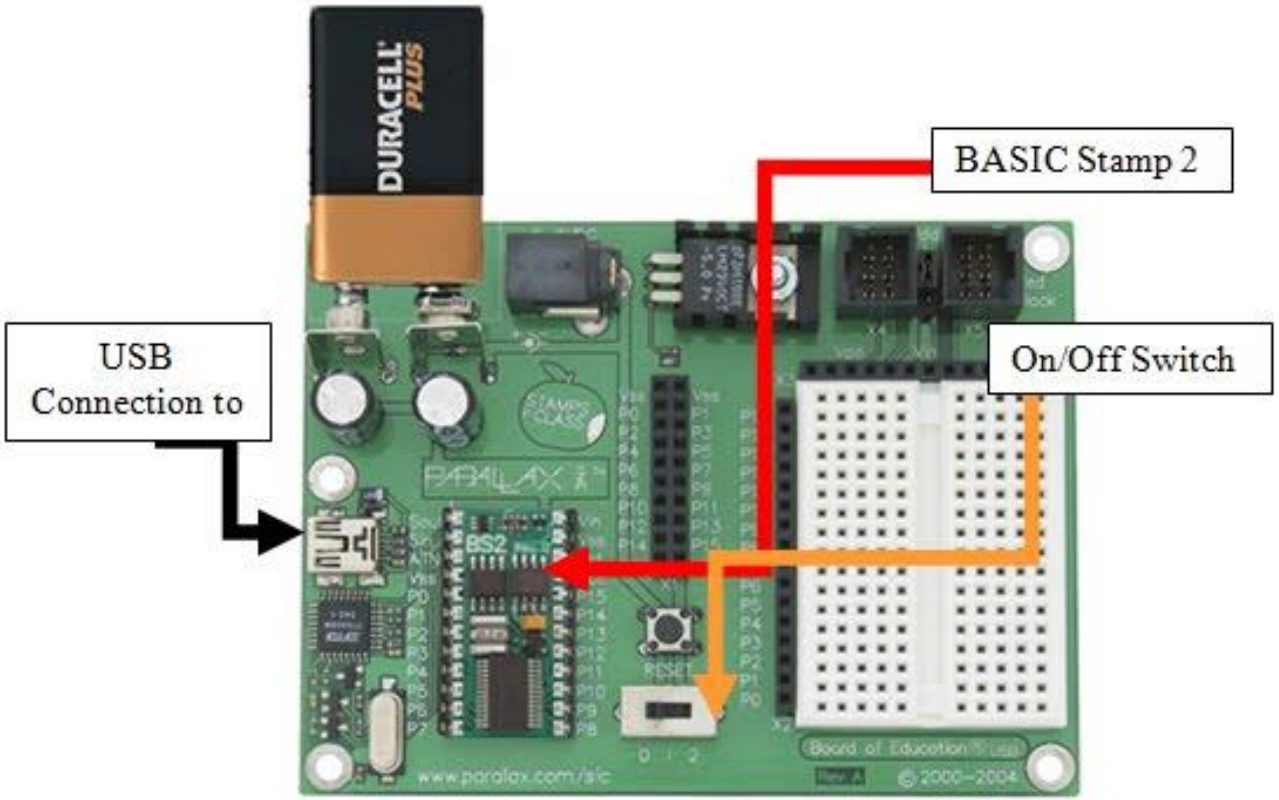
MCU: BASIC Stamp 2

- There are many version of BASIC Stamp MCU.

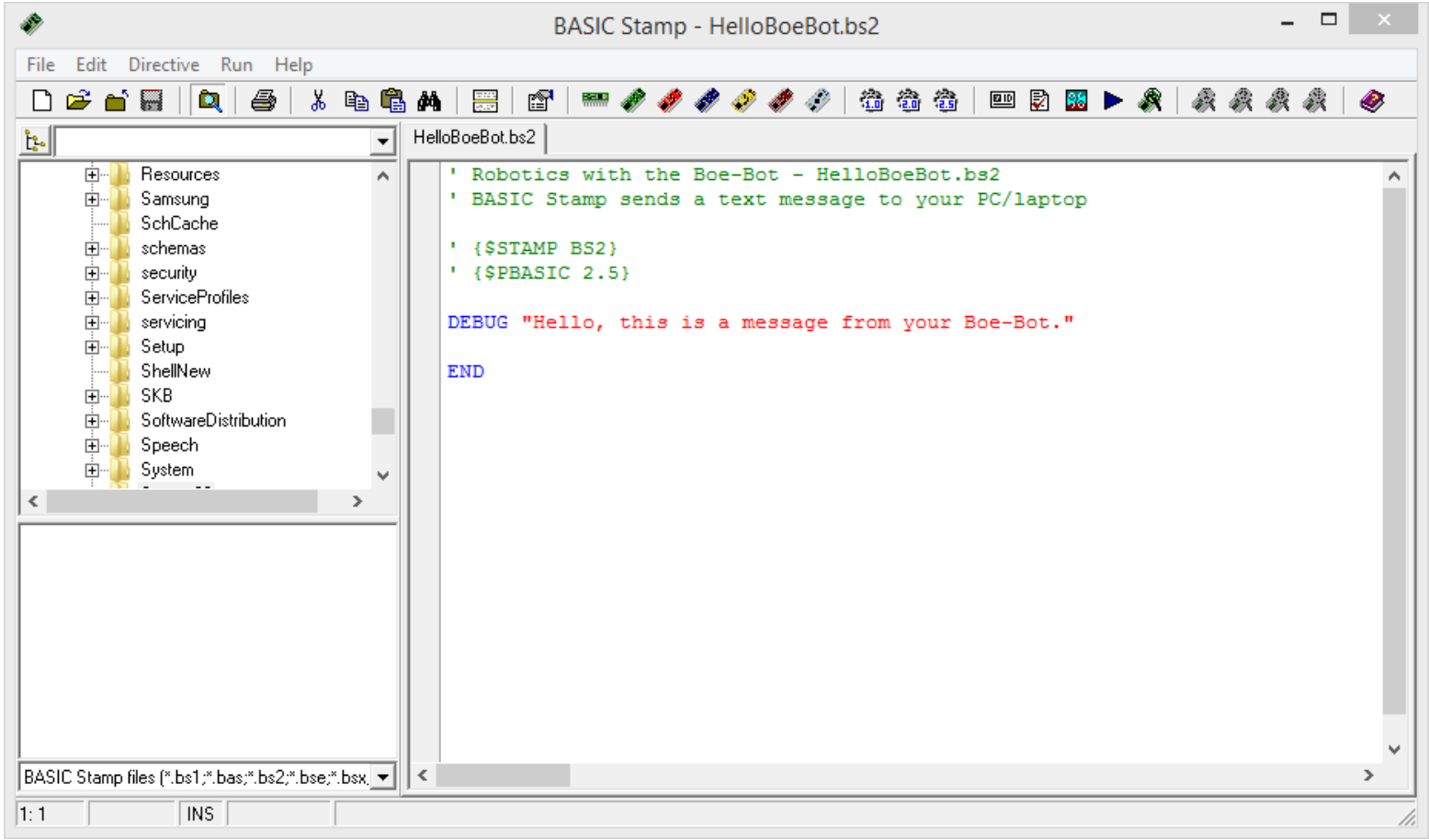




Programmer: Education Board



Software: BASIC Stamp Editor



Preparation

- Download BASIC Stamp Editor from Parallax website (or from Moodle):
  - https://www.parallax.com/sites/default/files/downloads/BS-Setup-Stamp-Editor-v2.5.3-%28r2%29.exe
- Install the BASIC Stamp Editor.

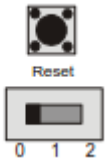
PROGRAMMING: ACTUATORS

BS2 Program Structure

- Always start with

```
' {$STAMP BS2}
' {$PBASIC 2.5}
```

- Add from toolbar
- Run from toolbar



(Switch to 1)

0 – OFF
1 – ON (no power to motors)
2 – ON (power to motors)

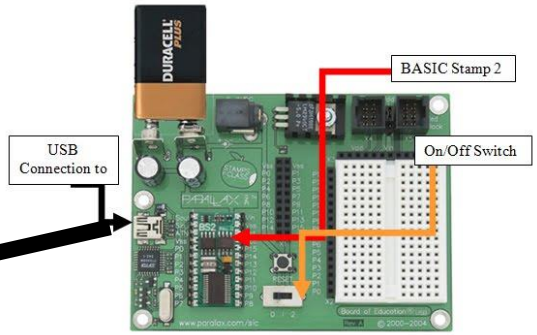
Messages from the Robot (MCU)

- Use of DEBUG **command** (instruction) for the MCU to communicate with PC (programmer).

```
DEBUG "Hello, this is a message from your Boe-Bot."
```

- **Exercise:** Try other messages.

(Programmer)



(Robot)

Breadboard: Prototyping

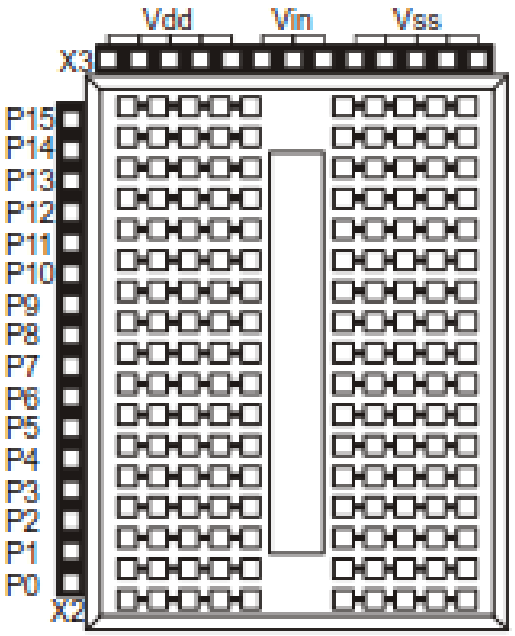


Figure B-2
Prototyping Area

Power terminals (black sockets along top), I/O pin access (black sockets along the side), and solderless breadboard (white sockets).

Resistors

Digit	Color
0	Black
1	Brown
2	Red
3	Orange
4	Yellow
5	Green
6	Blue
7	Violet
8	Gray
9	White

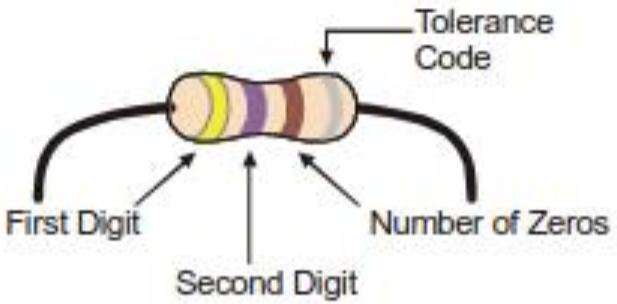


Figure B-1
Resistor Color Codes

- First stripe is yellow, which means leftmost digit is a 4.
- Second stripe is violet, which means next digit is a 7.
- Third stripe is brown. Since brown is 1, it means add one zero to the right of the first two digits.

Light Emitting Diode (LED)

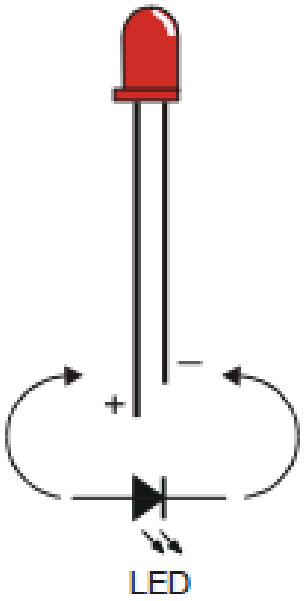


Figure 2-3
LED Part Drawing and Schematic Symbol

Part drawing (above) and schematic symbol (below)

The LED part drawings in later pictures will have a + next to the anode leg.

Turn ON LED

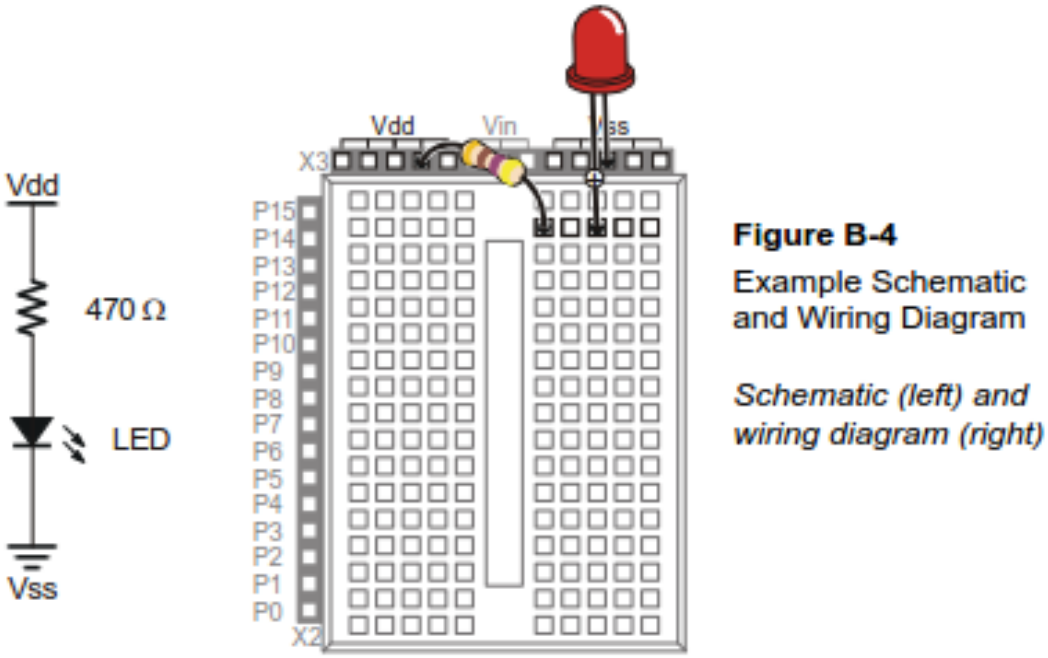


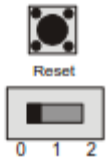
Figure B-4
Example Schematic and Wiring Diagram

Schematic (left) and wiring diagram (right)

Turn ON LED by MCU

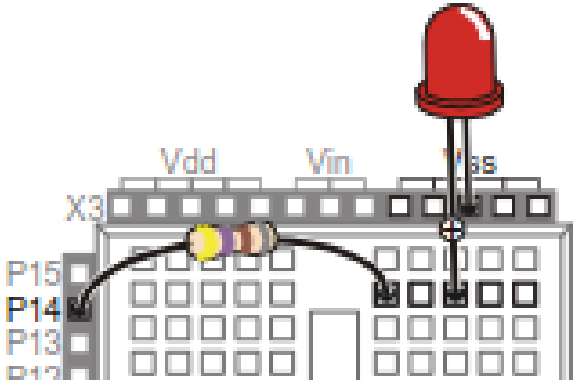
- Output PIN **commands**: HIGH and LOW.
- Exercise: Try to turn ON/OFF LED connected to PIN 14.

```
HIGH 14  'Send HIGH to PIN 14

LOW 14   'Send LOW to PIN 14
```



(Switch to 1)

- **Comments**: start with single quote (')
  - They are not commands, and will be ignored by MCU.



Time Delay

- PAUSE

```
PAUSE 1000  'Wait for 1000ms (1s)
```

- Exercise: Try

```
HIGH 13  'Send HIGH to PIN 13
LOW 13   'Send LOW to PIN 13
```

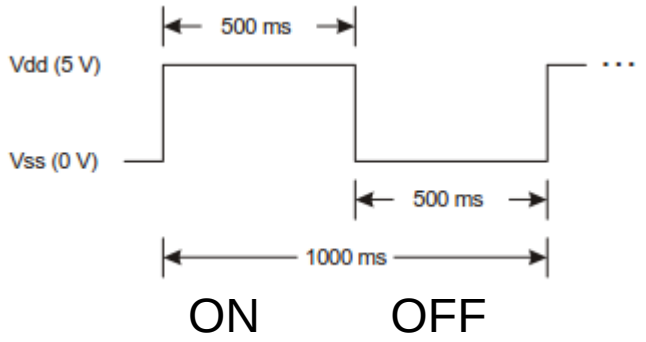
```
HIGH 13  'Send HIGH to PIN 13
PAUSE 1000  'Wait for 1000ms (1s)
LOW 13   'Send LOW to PIN 13
PAUSE 1000  'Wait for 1000ms (1s)
```

Keep Going: LOOP

- DO … LOOP
- Exercise: Try

```
HIGH 13  'Send HIGH to PIN 13
PAUSE 500  'Wait for 500ms (0.5s)
LOW 13   'Send LOW to PIN 13
PAUSE 500  'Wait for 500ms (0.5s)
```

```
DO
      HIGH 13  'Send HIGH to PIN 13
      PAUSE 500  'Wait for 500ms (0.5s)
      LOW 13   'Send LOW to PIN 13
      PAUSE 500  'Wait for 500ms (0.5s)
LOOP
```



Exercise

- DEBUG “Hello!” once every second

```
DEBUG "Hello!“, CR
```

Store Numbers: CON

- Convenient and flexible way of storing numbers: use constant.
- Exercise: Try

```
redLED CON 13 'the variable redLED store a constant number 13

DO
      HIGH redLED
      PAUSE 1000
      LOW redLED
      PAUSE 1000
LOOP
```

Exercise

- Try flash two LEDs at:
  - 1s ON, 1s OFF
  - 0.5s ON, 1s OFF
- You can play with different ON/OFF time.

Variables and Maths

- Similar to constant (CON), variables (VAR) are container for numbers (data).
- In contrast to constant (CON), content of variables (VAR) can be altered when program is running.
- Variables make dealing with numbers convenient.

13 + 3 = 16 vs

x = 13
y = 3
z = x + y

z = ?

Exercise: Try

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

value VAR Word              ' Declare variables
anotherValue  VAR Word

value = 500                 ' Initialise variables
anotherValue = 2000

DEBUG ? value               ' Display variables
DEBUG ? anotherValue

value = 10 * anotherValue

DEBUG ? value
DEBUG ? anotherValue

END
```

Note: Variable name usually starts with lower case.

Counting

- FOR … NEXT
  - Keep going for a fixed number of times.
- Exercise: Try

```
' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"

myCounter VAR Word

FOR myCounter = 1 TO 10
  DEBUG ? myCounter
  PAUSE 500
NEXT

DEBUG CR, "All done!"

END
```

Send Pulses

- Looping ON/OFF is equivalent to sending pulses.

PULSOUT Pin, Duration 'e.g. PULSOUT 13, ?

The Duration is in units of 2us. And it is limited to 16-bit, i.e. 65,535. Therefore, the longest duration is 131,070us = 131 ms.

PULSOUT Pin, Duration
The Duration is in units of 2us.

≡

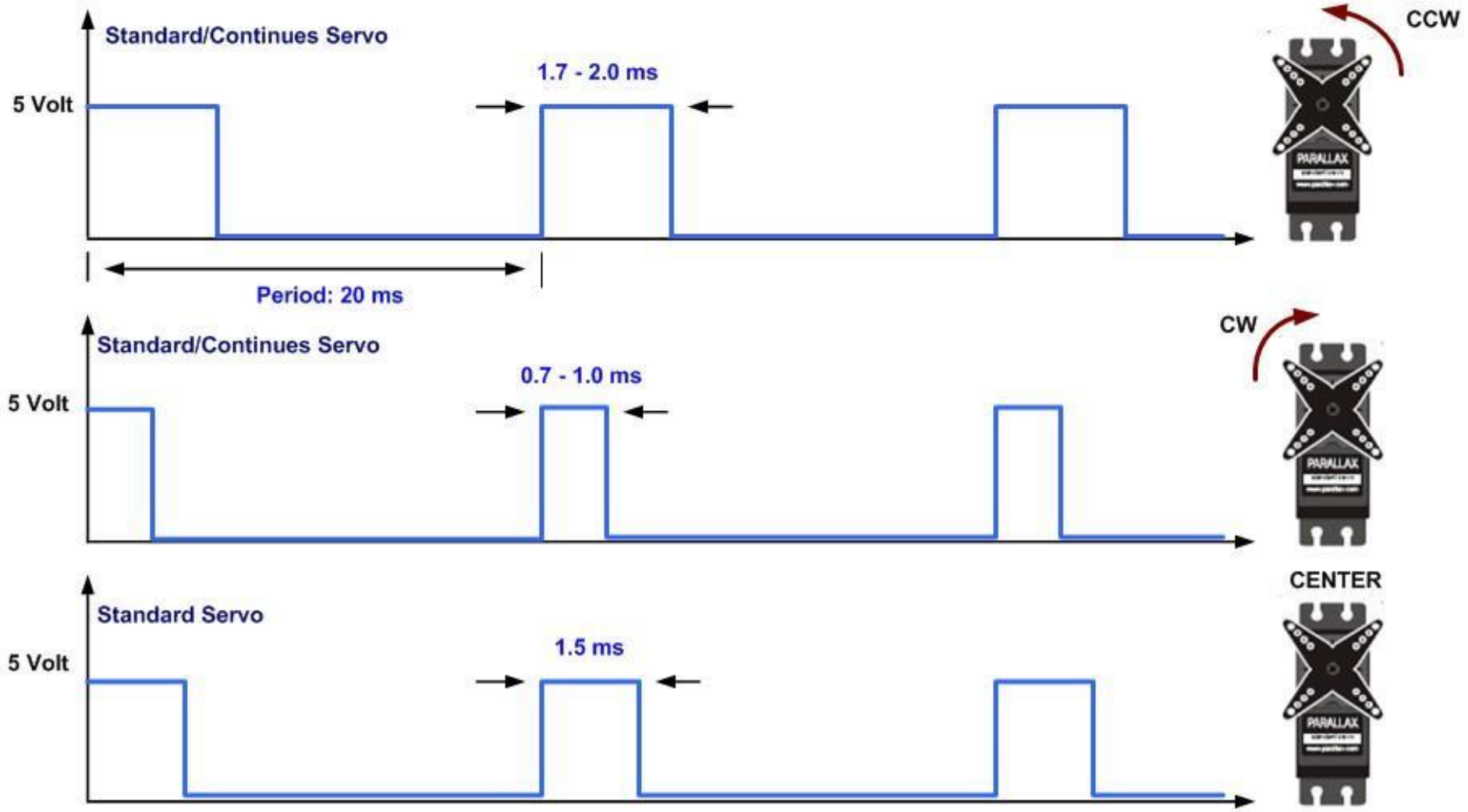
HIGH Pin
PAUSE Duration*
LOW Pin
The Duration is in units of 1ms.

- Exercise: Determine the duration value for PULSOUT for a 1.5ms pulse. Can you do a 1.5ms pulse using HIGH and LOW?

Servo Motor

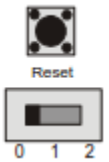
- PULSOUT will be useful to move a servo motor.
  - Servo motors are moved (controlled) by pulse width.
- Two types of servo motor:
  - **Standard**: pulse width control rotation angle.
  - **Continuous rotation**: pulse width control rotation speed.

Pulse Width Modulation (PWM)



Continuous Rotation Servo

- When first use, send a “neutral” or “stop” pulse width and tune motor to stop.
  - 1.5ms = 750 stop (each 1 is 2us)
  - 1.3ms = 650 full clockwise speed
  - 1.7ms = 850 full anticlockwise speed



(Switch to 2)

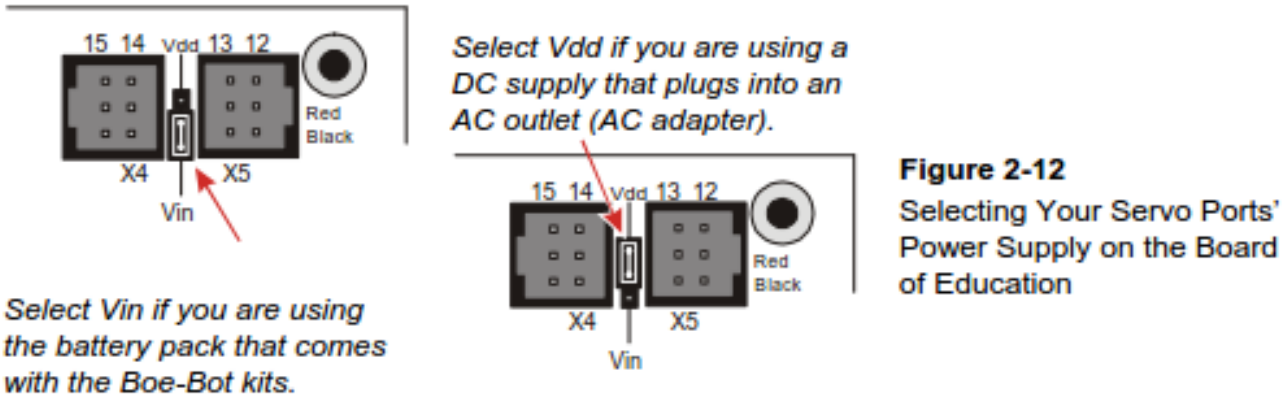
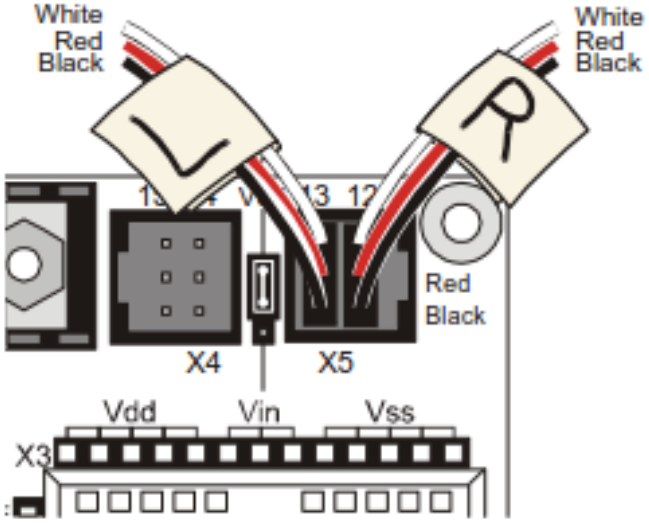


Figure 2-12
Selecting Your Servo Ports' Power Supply on the Board of Education

PULSOUT to Control Servo

- Connect two Continuous Rotation servos as below:

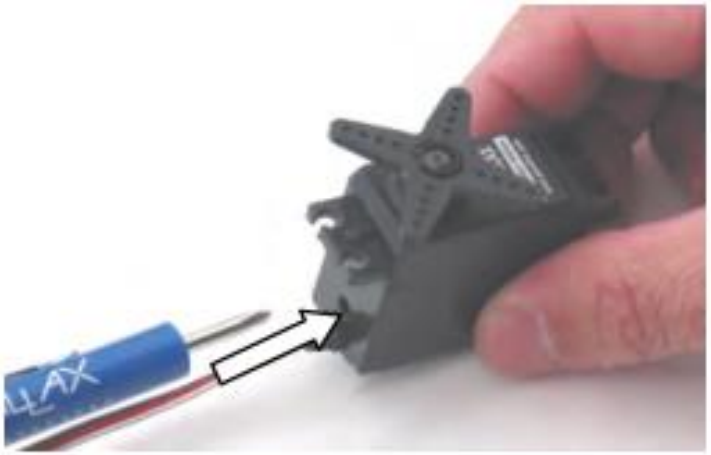



```
DO
   PULSOUT 12, 750
   PAUSE 20
LOOP
```

- Note the motors are now connected to P12 and P13.
- Exercise: Send a “stop (750)” signal to each motor. If the motor(s) is turning, “tune” it to stop (see next slide).

Tune (Center) a Servo

- Do this very slowly to get the servo stop, when sending a “stop” signal.

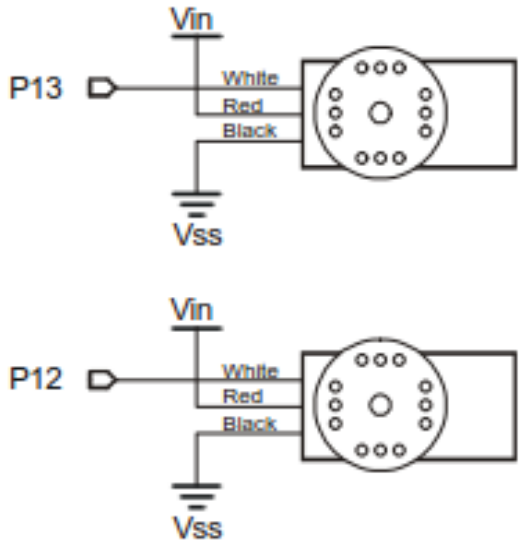


1) Insert tip of Phillips screwdriver into potentiometer access hole.

2) Gently turn screwdriver to adjust potentiometer until the servo stops moving.

Exercises

- Center servo
- Rotate servo clockwise, anticlockwise, change speed
- Rotate two servos, in opposite directions, in same directions.



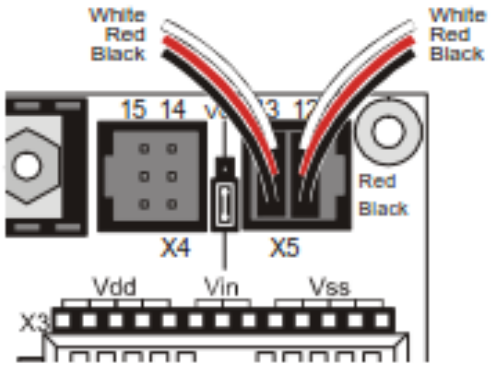


Figure 2-13 Servo Connections for the Board of Education

Standard Servo

- They are controlled in exact same way as the continuous rotation servo.
  - 1.5ms = 750 neutral position
  - 0.4ms = 200 full clockwise (check the angle)
  - 2.4ms = 1200 full anticlockwise (check the angle)
  - The above values are dependent on individual servo. If the servo is vibrating, then you have exceeded its limit.

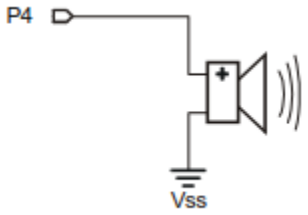
```
counter VAR Word

FOR counter = 1 TO 220
    PULSOUT 12, 750
    PAUSE 20
NEXT
```

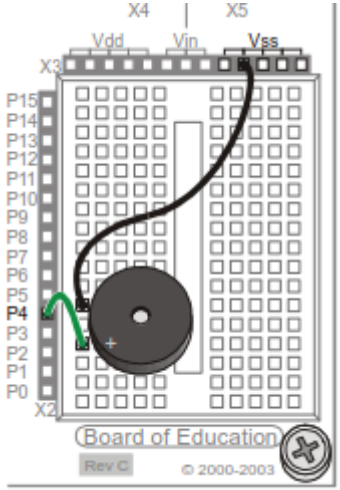
Note: without the pulse, the servo is “free”. Depending on application, to “hold” the servo in a position, we need to loop (FOR … NEXT or DO … LOOP).

Another actuator: Buzzer

- The buzzer will buzz when we send a continuous pulse (with same ON/OFF duration) at audible frequency.

```
FREQOUT Pin, Duration, Frequency
```





- Exercise: Try different frequency.
  - Audible frequency: 20Hz to 20 kHz (for perfect ears)

```
FREQOUT 4, 2000, 3000 'send frequency 3000 Hz for 2000 ms to PIN 4
```

Quick Sum Up

- By now you know how to program three types of actuators (outputs):
  - LEDs, Servo motors, Buzzers.
  - DEBUG from MCU to PC (programmer, human).
- In addition to a few programming concepts:
  - Arranging sequence of instructions.
  - Program flow control: infinite loop, for loop.
  - Program structure: start of the program.
  - Program syntax: the commands, the comments.
  - How to deal with the data: variables, constant.