Sonar Car 1 – Servo Positioning
Cornerstone Electronics Technology and Robotics II

- **Administration:**
  - Prayer
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
  - Lab 1 servo1 as .pdf file: http://www.cornerstonerobotics.org/code/servo1.pdf
- **Introduction:**
  - **Obstacle Avoidance:** This series of four sonar car lessons walk you through the software development for the sonar car. In general, the car is designed to make its way through an obstacle without hitting an obstacle or wall. The sonar car takes 7 ultra-sonic distance readings, chooses the longest distance reading of the seven, and then rotates the car in the direction of the longest distance reading before traveling forward. For example in Figure 1, the sonar car should determine that direction 4 is the longest distance reading at its current position and it should rotate in that direction and then move forward.

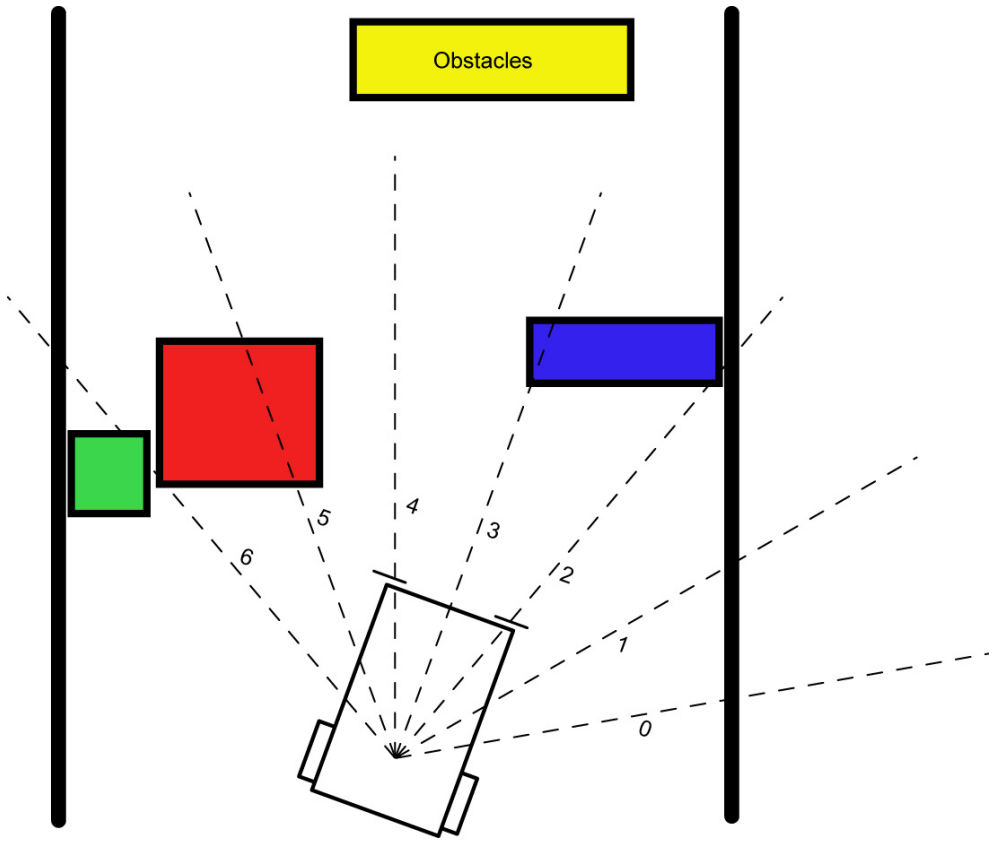


Figure 1

PicBasic Pro Review:

- **PULSOUT:**

  Format:

  PULSOUT *Pin*, *Period*

  Explanation:
  Generates a pulse on *Pin* of specified *Period*. The pulse is generated by toggling the pin twice, thus the initial state of the pin determines the polarity of the pulse. *Pin* is automatically made an output. *Pin* may be a constant, 0 - 15, or a variable that contains a number 0 - 15 (e.g. B0) or a pin name (e.g. PORTA.0). The resolution of **PULSOUT** is dependent upon the oscillator frequency. If a 4MHz oscillator is used, the *Period* of the generated pulse will be in 10us increments. If a 20MHz oscillator is used, *Period* will have a 2us resolution. Defining an OSC value has no effect on **PULSOUT**. The resolution always changes with the actual oscillator speed.

Examples:

```
PULSOUT PORTB.5,100   ' Send a pulse 1 msec long (at 4MHz) to RB5

PULSOUT 2,200         'Send a pulse 2 msec long to RB2.
```

- **FOR..NEXT:**

  Format:

  FOR *Count* **=** *Start* **TO** *End* {**STEP** {**-**} *Inc*}
  {*Body*}
  NEXT {*Count*}

Explanation:
The **FOR**..**NEXT** loop allows programs to execute a number of statements (the *Body*) some number of times using a variable as a counter. Due to its complexity and versatility, **FOR**..**NEXT** is best described step by step:
1) The value of *Start* is assigned to the index variable, *Count*. *Count* can be a variable of any type.
2) The *Body* is executed. The *Body* is optional and can be omitted (perhaps for a delay loop).
3) The value of *Inc* is added to (or subtracted from if "**-**" is specified) *Count*. If no **STEP** clause is defined, *Count* is incremented by one.
4) If *Count* has not passed *End* or overflowed the variable type, execution returns to Step 2.
If the loop needs to *Count* to more than 255, a word-sized variable must be used.
Examples:

```
FOR i = 1 TO 10      ' Count from 1 to 10
   N = N+1
NEXT i               ' Go back to and do next count
```

```
FOR B2 = 200 TO 100 STEP -1  ' Count from 200 to 100 by -1
  PULSOUT 2,B2               ' Send position signal to servo
  PAUSE 20                   ' Pause 20 msec
NEXT B2                      ' Go back to and do next count
```

Nesting:
A **For..Next** loop can be placed inside another **FOR..NEXT** loop; this is called nesting. Nesting is embedding one object (one **FOR..NEXT** loop) in another object of the same type (another **FOR..NEXT** loop).

```
FOR i = 1 TO 2

    FOR j = 1 TO 10
        HIGH 0
        LOW 1
        PAUSE 500
        LOW 0
        HIGH 1
        PAUSE 500
    NEXT j

    HIGH 2
    PAUSE 500
    LOW 2

NEXT i
```

Note: The FOR…NEXT j loop must be nested entirely within the FOR…NEXT i loop.

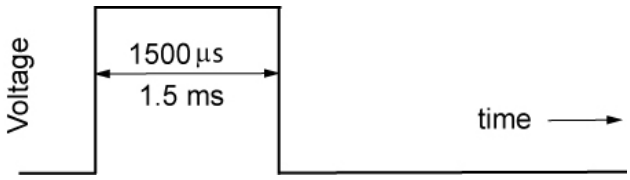
- **Power Supplies for Digital Circuits Review:**
  - Digital circuits require power supplies that provide a stable source of power to all circuit components.
  - Even though our circuits are powered from a voltage regulator, motors and servos can disrupt the proper operation of the PIC microcontroller.
  - **Make certain that the servo power supply is separate from the PIC power supply, i.e., have two +5V regulated power supplies.** Otherwise, if the servo spikes the single power line supplying power to both the servo and the PIC, the 16F88 may reset.
- **Servos Review:**
  - A servo is a special motor used to provide control for a desired rotational position through the use of feedback.
  - Most servos have a position range of 180 degrees.
  - A hobby servo is typically used to provide actuation for various mechanical systems such as the steering of a RC car, the flaps on a RC plane, or the rudder of a RC boat.
- **Servo Connections:**

  Black = Ground
  Red = +5 to +6 dc volts
  White/Orange = Control wire (Futaba uses a white wire)

- **Servo Pulse Signal Widths:**
  - Pulse width for full counterclockwise position:

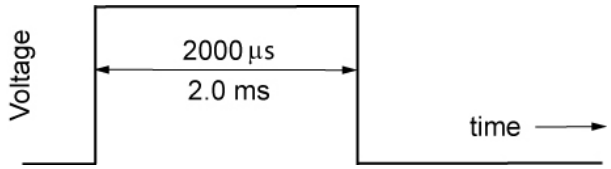
0.9 ms Pulse Width

  - Pulse width for center position:



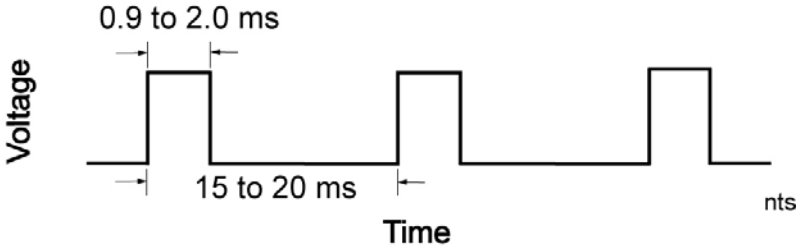
1.5 ms Pulse Width

  - Pulse width for full clockwise position:



2.0 ms Pulse Width

  - Servo Motor Pulse Intervals:
    - The period of one servo pulse cycle is typically 20 ms.



  - Perform LAB 4 – servo1, servo2, servo3, and servo4.

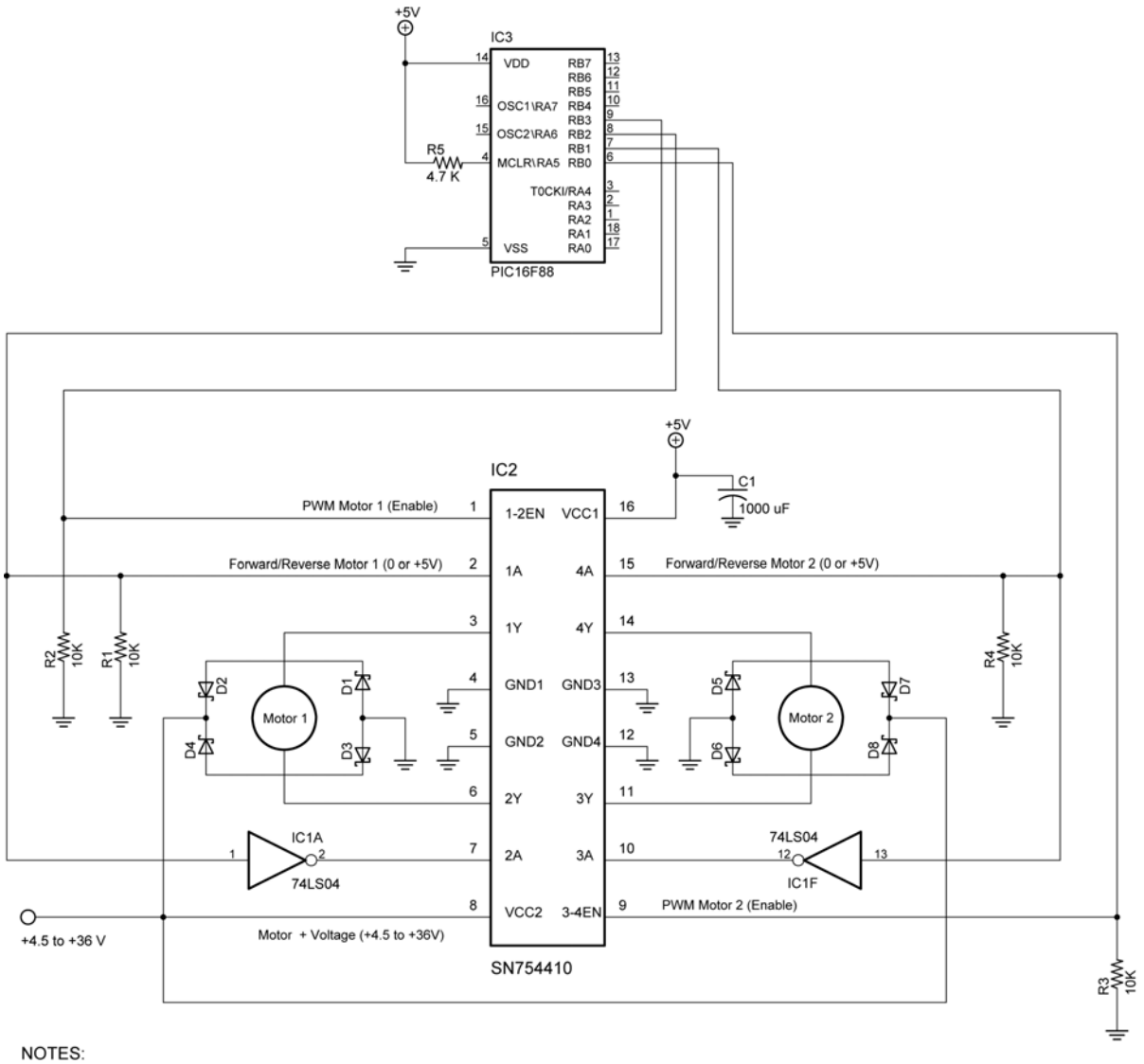
Sonar Car 1 – Servo Positioning
LAB 1 – Set Servo into Starting Position
Cornerstone Electronics Technology and Robotics II

- **Purpose:** The purpose of this lab is to set the servo on the sonar car into the starting position.

- **Apparatus and Materials:**
    - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics below.

- **Procedure:**
    - You can import **servo1.pbp** to provide the basic program code. You will need to make modifications to this program. See: http://cornerstonerobotics.org/code/servo1.pbp.
    - Save the imported program as **sonar_car1.pbp**. You will keep adding to this program in the coming weeks until your sonar car is fully functional.
    - Give the output pin PORTB.6 the name servo_pin. It can be assigned a name using the **VAR** command. This pin will send the program signal to the servo.
    - Using the **PULSOUT** command, set the Period to 70. Remember the format for **PULSOUT** is: **PULSOUT** *Pin, Period*
    - Shorten the time to move the servo by reducing the **FOR..NEXT** loop to 20 loops.

Sonar Car Circuitry 1

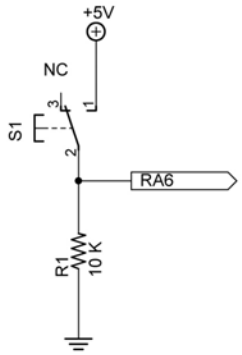


NOTES:

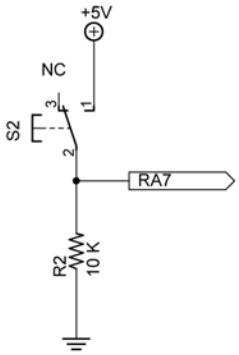
1. D1 - D8 1N5817
2. 74LS04 Pin 7 GND
   74LS04 Pin 14 Vcc +5V
3. SN754410 GNDS Also Act As
   Heatsinks, Ground Separately

Texas Instrument SN754410 H-Bridge Motor Driver

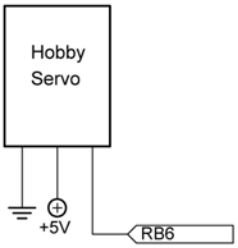
Sonar Car Circuitry 2



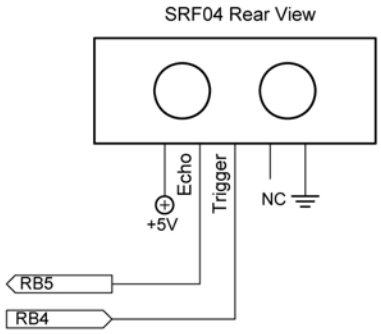
Left Bumper Switch



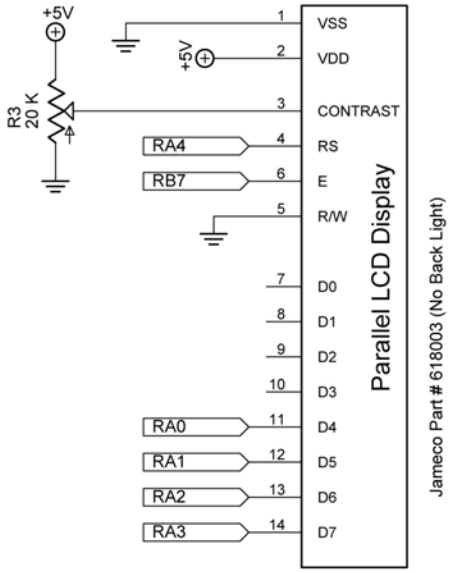
Right Bumper Switch



Servo



Ultrasonic Range Finder



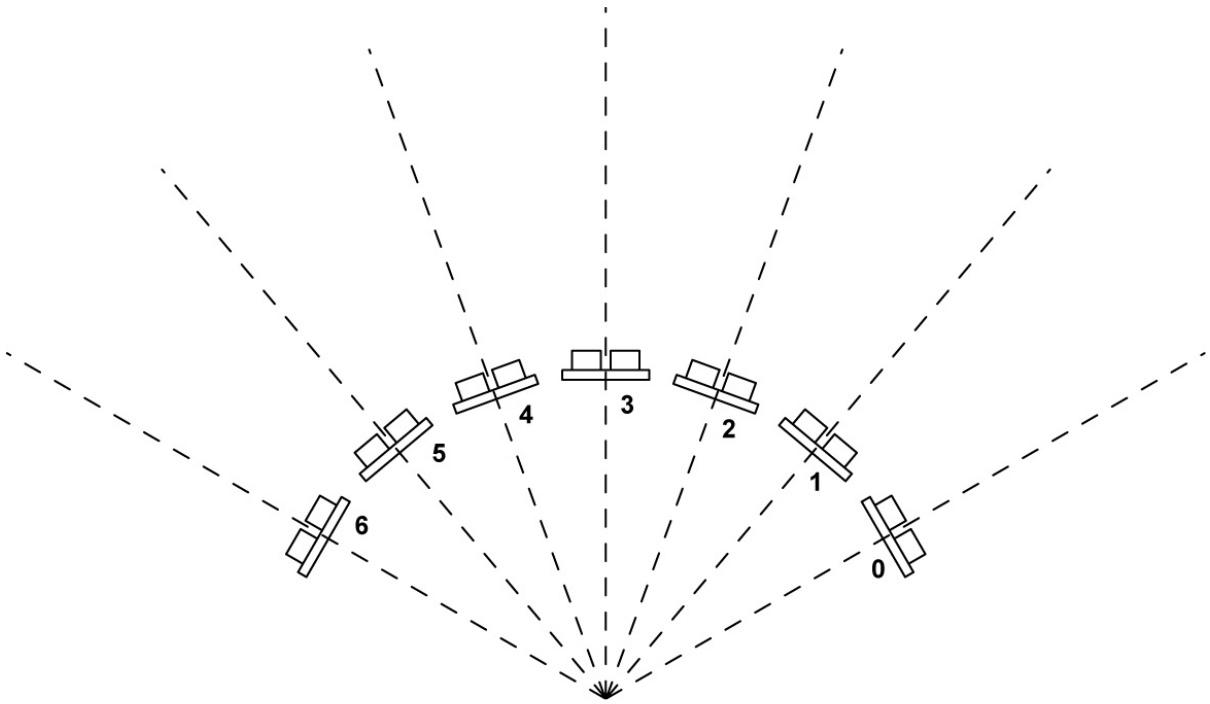
LCD

Sonar Car 1 – Servo Positioning
LAB 2 – Pan Servo across the Front on the Sonar Car
Cornerstone Electronics Technology and Robotics II

- **Purpose:** The purpose of this lab is to make the servo pan across the front of the sonar car in six discrete steps.
- **Apparatus and Materials:**
  - 1 – Sonar car with Sonar Car Circuitry 1 & 2 on breadboard – see schematics in Lab 1.
- **Procedure:**
  - Declare the following variables

```
p0      VAR   BYTE      ' BYTE to store servo pulse period
c0      VAR   WORD      ' WORD for counter
```

  - Start the servo at the position generated by Period = 70 (see Lab 1) and then rotate through to Period = 208 in 6 discrete steps. Hint: Nest two **FOR..NEXT** loops.
  - Shorten the time to move the servo to each new position by reducing the **FOR..NEXT** loop to 15 loops.
  - Set up a loop to make the servo return to the starting position and then pan across the front of the car again. The



7 Servo Positions with Ultra-sonic Range Finder

  - Compare your program to **sonar_car_a.pbp**. See: http://cornerstonerobotics.org/code/sonar_car_a.pbp