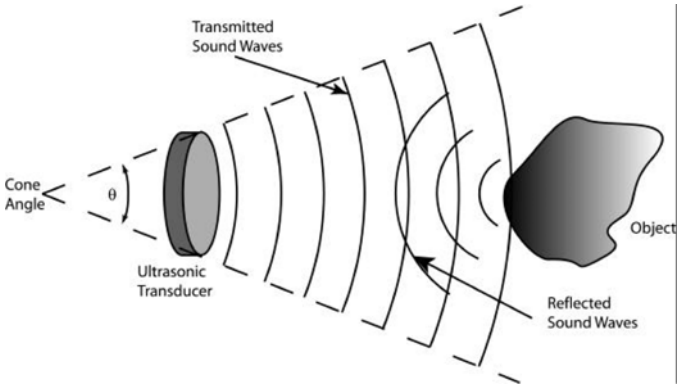
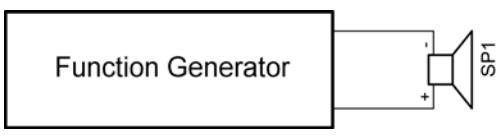
Devantech SRF04 Ultra-Sonic Ranger Finder
Cornerstone Electronics Technology and Robotics II

- **Administration:**
  - Prayer
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
  - Lab 1 array1.pdf: http://www.cornerstonerobotics.org/code/array1.pdf
  - Lab 1 array2.pdf: http://www.cornerstonerobotics.org/code/array2.pdf
  - Lab 2 sonar1.pdf: http://www.cornerstonerobotics.org/code/sonar1.pdf
  - Lab 3 sonar_car1.pdf: http://www.cornerstonerobotics.org/code/sonar_car1.pdf
- **SRF04 Ultra-Sonic Sensor:**
  - Introduction**:**
    - Gives precise non-contact distance measurements
    - Measures distances from 3 cm (1.2”) to 3 m (39.5”)
    - Generates an ultrasonic sound burst, or ping, and then measures the time it takes for the echo to return to the receiver. See figure below:



Sonar Sensor Function From:
http://www.mech.utah.edu/~me3200/labs/F03Labs/F03_Ultrasonic_L7.pdf

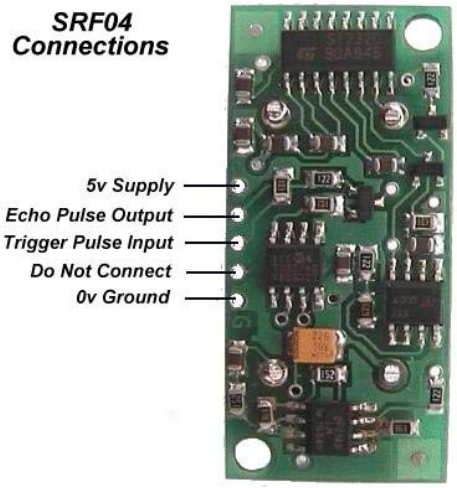
- Ultrasonic means the frequency of the sonic (sound) pulse is above the human hearing range. The highest frequency that is detectable by the human ear is approximately 20 KHz.
  - Using a function generator and the following circuit, test the 20 KHz limit.



Sound Generator Circuit

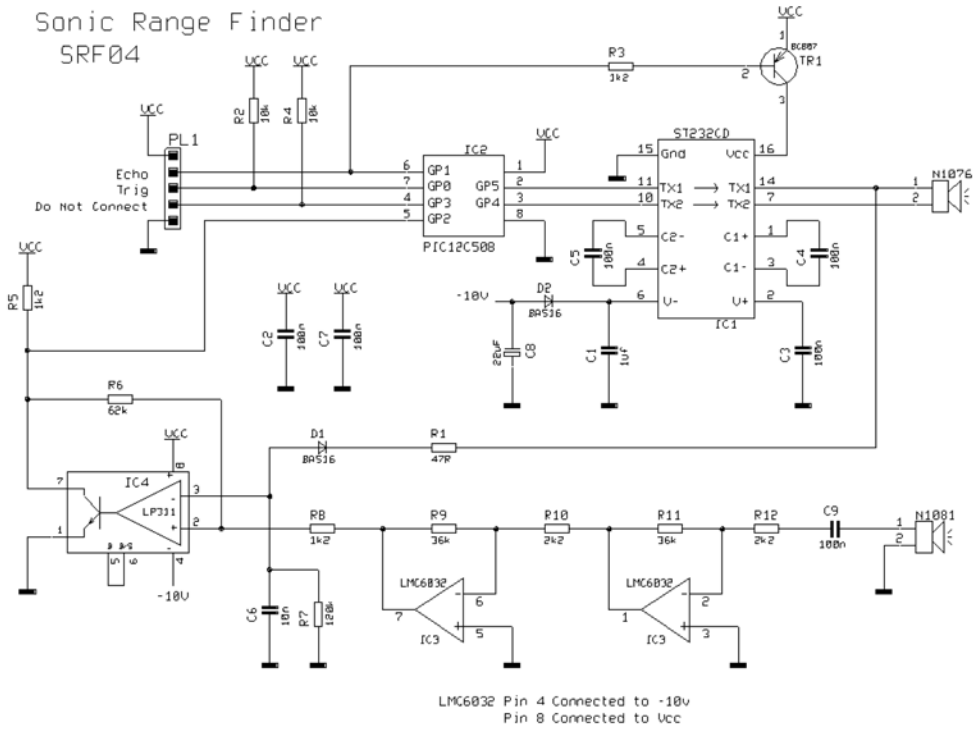
        - The SRF04 Ultra-Sonic ranger finder frequency is 40 kHz.
    - The ultrasonic pulse travels at the speed of sound (1087 ft/sec or 1.087 ft/msec). The speed of sound varies with temperature, humidity and altitude.
    - The output from the SRF04 is a variable width pulse from 100 usec to 18 msec depending upon the distance to the object detected (the target).
    - The SRF04 has a separate transmitter and receiver transducer while some sonar sensors have but a single transducer.
- Robotic Uses:
    - Navigation, obstacle avoidance
    - Distance measurements
- Other Uses:
    - Auto-focusing cameras
- Better performance than IR when:
    - High ambient infrared light levels, such as bright sunlight
    - Encountering dark objects that do not reflect the IR energy
- Specifications:
    - Voltage: 5 vdc
    - Current: 30 mA typical, 50 mA maximum
    - Frequency: 40 KHz
    - Minimum Range: 3 cm
    - Maximum Range: 3 m
    - Input Trigger: 10 usec minimum, TTL level pulse
    - Echo Pulse: Positive TTL level signal with the width proportional to the range of the object
    - Dimensions: 43 mm long x 20 mm wide x 17 mm high
- Operation:
    - The user sends a 10 usec trigger pulse to the SRF04 module.
    - The user trigger pulse causes the ultrasonic ranger to send out a burst of 8 sonic pulses at 40 KHz.
    - The trigger pulse also activates the echo receiver which awaits an echo pulse.
    - If an echo is received, the SRF04 module outputs an echo pulse whose width is proportional to the distance to the object detected.

- SRF04 Connections:



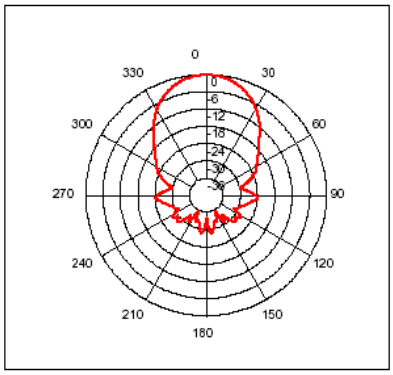
SRF04 Connections
From: http://www.robot-electronics.co.uk/htm/srf04tech.htm

- SRF04 Schematic:



SRF04 Schematic
From: http://www.wwwactive-robots.com/products/sensors/devantech/srf04_sonar_schematic.pdf

- Beam Pattern:



SRF04 Beam Pattern
From: http://www.robot-electronics.co.uk/htm/srf04tech.htm

- Mounting:
  - If you mount the SRF04 module lower than 12" above the floor, point it slightly upwards to avoid reflections from the flooring material.
- Using more than one SRF04 range finder at one time:
  - In our class, do not "fire" two SRF04 modules at one time since they will pick up each other's "ping" and result in a false reading. Fire them sequentially 65 msec apart.

- **Arrays:**
  - Definition for our purposes: An array is a rectangular arrangement of quantities in rows and columns, as in a matrix.
  - Our application - variable arrays:
    - This lesson will use a variable array to track the first variable - several servo positions and the second variable – the respective sonar readings from a SRF04 range finder.
  - Array format in PicBasic Pro:
    - Variable arrays are created or declared as follows:

      Label **VAR** Size**[**Number of elements**]**

      Where:
      - Label is the name of the array (excludes keywords)
      - Size is **BIT, BYTE, or WORD**
      - Number of elements is the number of locations reserved in memory for the array
    - Examples:

      temp1 **VAR** **BYTE**[10]
      c0 **VAR** **BIT**[4]
      distance **VAR** **WORD**[8]

- In the first array, temp1[ ], the first element is temp1[0], the second element is temp1[1]. Since there are 10 elements in the array, so the last element is temp1[9]. See excel file below:
- Since memory is reserved for variable arrays like any other variable, there are size limits for each type. For the PIC16F88 the maximum number of elements for each type is as follows:

  BIT 768
  BYTE 96
  WORD 48

  The complier will not compile successfully unless the array will fit into the memory. The maximum number of elements for other PIC microcontrollers may be determined by changing the **[**Number of elements**]** when creating the array:

  Label **VAR** Size**[**Number of elements**]**

  For example, the PIC16F88 will not compile the following array creation since it exceeds the maximum number of elements for **BIT**.

  temp1 VAR **BIT**[769]

- Perform Ultra-Sonic Lab 1 – array1 and array2.pbp

- **New PicBasic Pro Commands:**
  - **SELECT CASE:**
    Format:

    SELECT CASE Var
    CASE Expression 1
    Statement
    CASE Expression 2
    Statement
    CASE Expression 3
    Statement
    ……
    CASE ELSE
    Statement
    END SELECT

    Explanation:
    SELECT CASE statements are used instead of multiple **IF..THEN** statements. Each **CASE** compares the value of the variable to the expression in that **CASE**. When the expression is true, the statement after that **CASE** is executed. When none of the **CASES** are true, the statement after the **CASE ELSE** is executed.

If the comparison in the expression is something other than equal, **IS** must be used.

Examples:

```
SELECT CASE x
        CASE 0
                HIGH  PORTB.0
        CASE 1
                HIGH  PORTB.1
        CASE 2
                HIGH  PORTB.2
        CASE ELSE
                GOTO loop
END SELECT
        PAUSE 500
        PORTB = 0

SELECT CASE dist
        CASE IS < 20
                GOSUB left_turn
        CASE IS > 30
                GOSUB right_turn
        CASE ELSE
                GOSUB straight
END SELECT
```

If the variable dist (for distance) is less than 20, the program will execute the subroutine left_turn. If the variable dist is greater than 30, the program will execute the subroutine right_turn.
When neither of the two **CASES** are true, i.e., 20 >= dist <= 30, the program executes the subroutine straight.

- **PULSIN:**

Format:

PULSIN *Pin*,*State*,*Var*

Explanation:
Measures pulse width on *Pin*. If *State* is zero, the width of a low pulse is measured. If *State* is one, the width of a high pulse is measured. The measured width is placed in *Var*. If the pulse edge never happens or the width of the pulse is too great to measure, *Var* is set to zero. If an 8-bit variable is used, only the LSB of the 16-bit measurement is returned. *Pin* is automatically made an input. *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 **PULSIN** is dependent upon the oscillator frequency. If a 4MHz oscillator is used, the pulse width is returned in 10us increments. If a 20MHz oscillator is used, the pulse width will have a 2us resolution. Defining an OSC

value has no effect on **PULSIN**. The resolution always changes with the actual oscillator speed.
PULSIN normally waits a maximum of 65535 counts before it determines there is no pulse. If it is desired to wait fewer counts before it stops looking for a pulse or the end of a pulse, a **DEFINE** can be added:

```
DEFINE PULSIN_MAX 1000
```

This **DEFINE** also affects **RCTIME** in the same manner.
Example:

```
PULSIN PORTB.4,1,W3  ‘ Measure high pulse on Pin4 stored in W3
```

- Perform Ultra-Sonic Lab 2 – sonar1.pbp
- Perform Ultra-Sonic Lab 3 – Completion Sonar Car

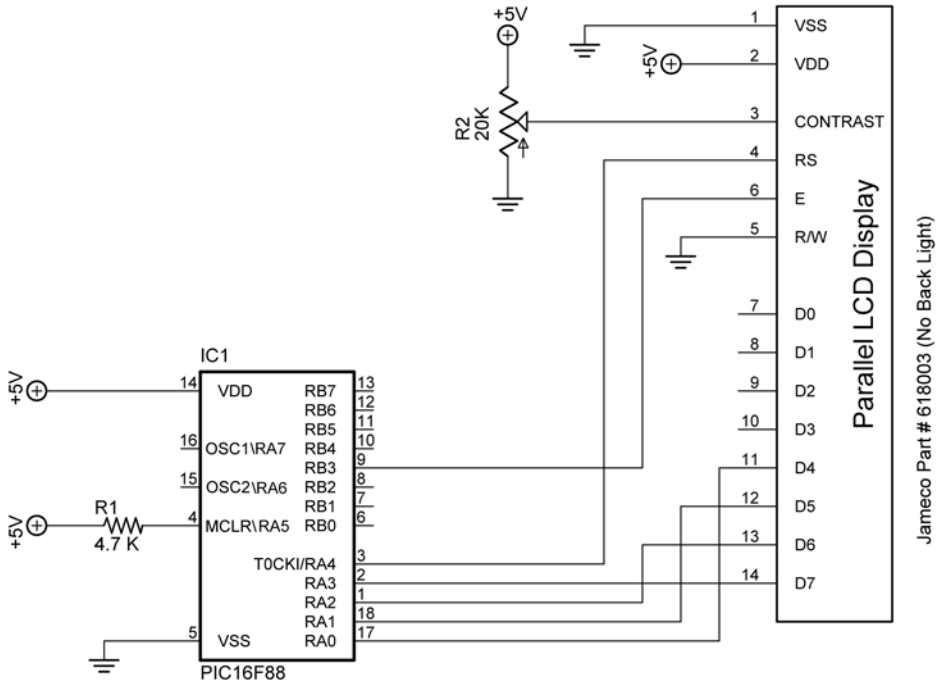
Cornerstone Electronics Technology and Robotics II
Ultra-Sonic LAB 1 – array1 and array2.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the PicBasic Pro variable arrays.

- **Apparatus and Materials:**

  - 1 – Breadboard
  - 1 – PIC 16F88 Microcontroller
  - 1 – 4.7K Ohm Resistor
  - 1 – 20K Tripot
  - 1 – LCD Screen, Jameco # 618003

- **Procedure:**
  - Wire the circuit array1 as shown below.
  - Program the PIC16F88 with **array1.pbp** and power the chip.
  - Program the PIC16F88 with **array2.pbp** and power the chip.
  - Program the PIC16F88 to three take CdS photoresistor readings and record the sample number and CdS reading in two arrays, sample[3] and cds[3]. Display all of the results of the two arrays at one time on an LCD. Save the program as **array10.pbp**.



array1 and array2

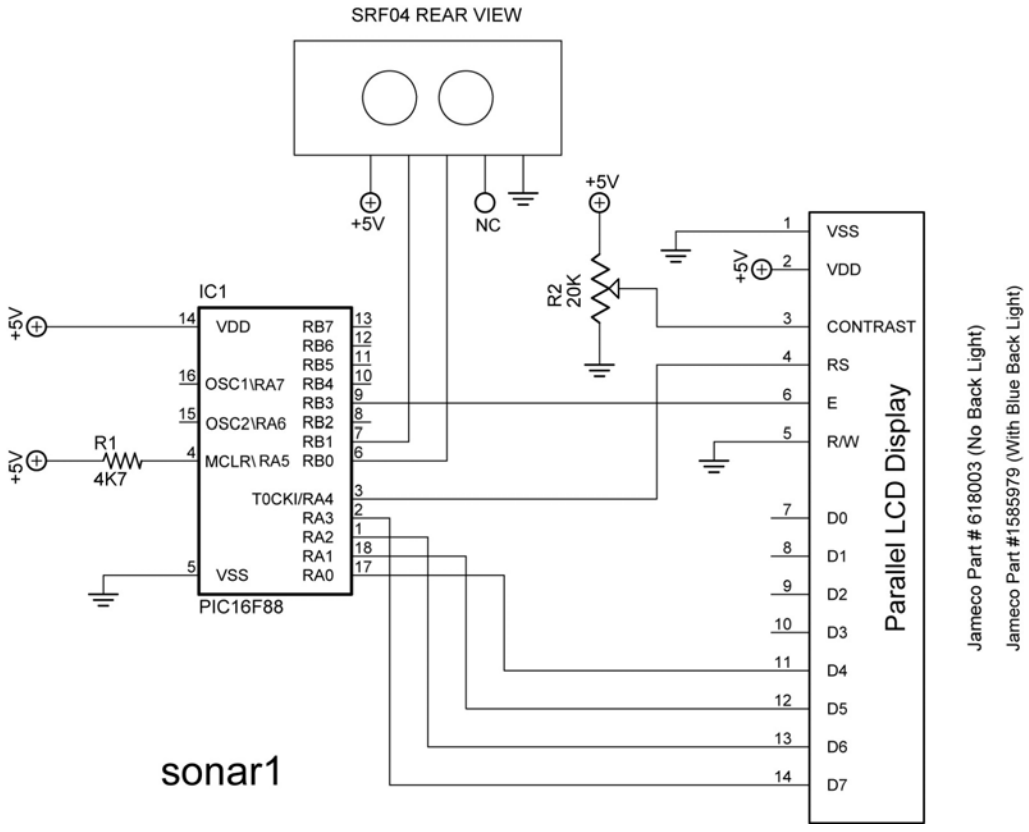
Cornerstone Electronics Technology and Robotics II
Ultra-Sonic LAB 2 – sonar1.pbp

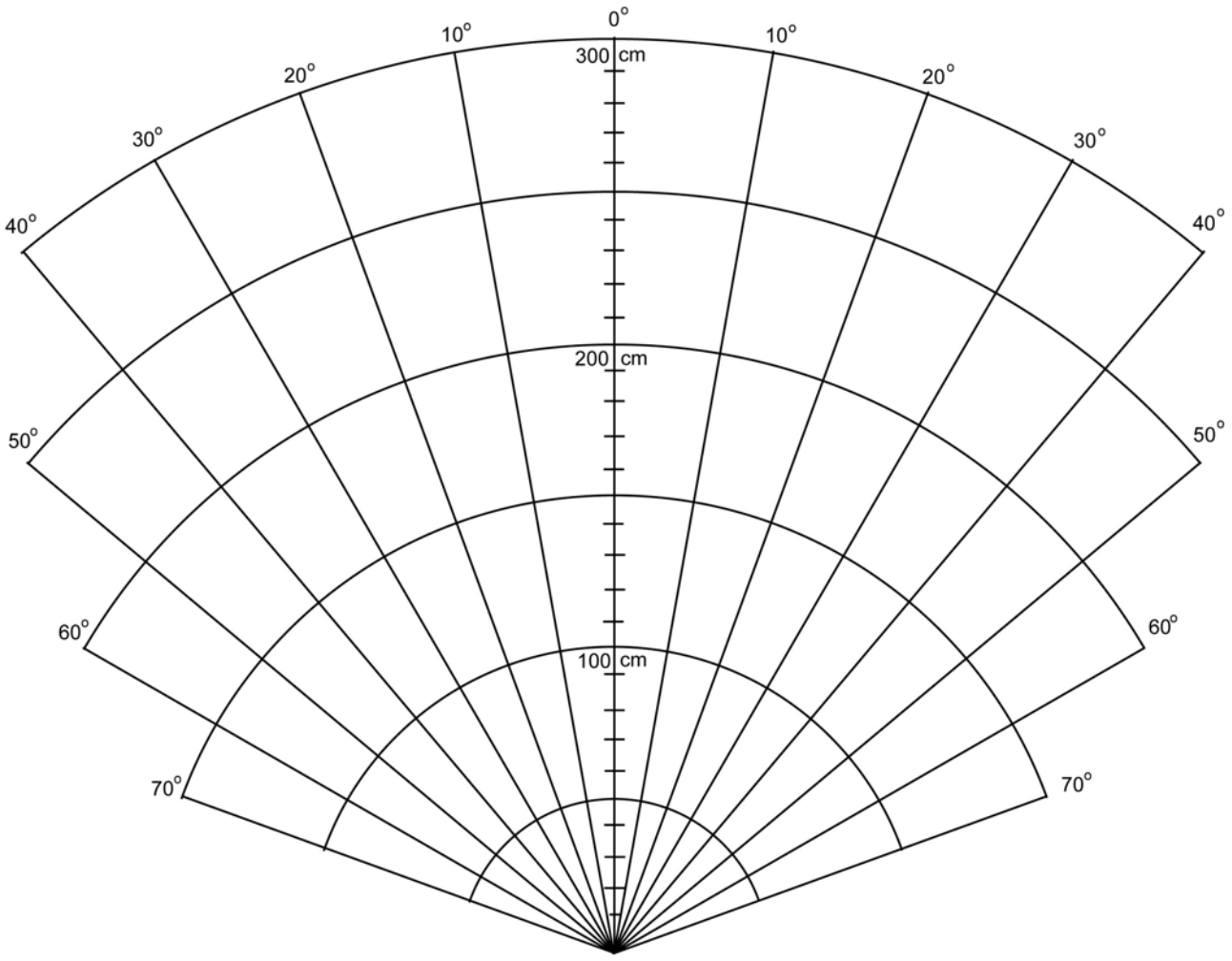
- **Purpose:** The purpose of this lab is to acquaint the student with the basic function of the SRF04 ultra-sonic range finder and the PicBasic Pro commands to drive the range finder.

- **Apparatus and Materials:**

    - 1 – Breadboard or Robotic Car
    - 1 – PIC 16F88 Microcontroller
    - 1 – 4.7K Ohm Resistor
    - 1 – 20K Tripot
    - 1 – LCD Screen, Jameco # 618003

- **Procedure:**
    - Wire the circuit sonar1.
    - Open **sonar1.pbp** and download to your chip.
        - Place an object about 10 inches from the sonar and let the sonar take readings over time. Observe any changes in the readings.
        - Use the attached SRF04 Beam Pattern Plot sheet to plot the sensitivity of the SRF04 module detecting a 2”x 4” piece of lumber. Keep the wood perpendicular to the radial lines that converge at the SRF04. Record only readings that are valid and consistent with the range of the sonar.





SRF04 Beam Pattern Plot

- **Challenge:**
  - Using a servo and a SRF04 for obstacle avoidance:
    - Use a SRF04 mounted on a servo to sweep through 180 degrees and navigate your robot to avoid obstacles. At this point, do not use an interrupt, but rather use pauses in your forward movement.

Cornerstone Electronics Technology and Robotics II
Ultra-Sonic LAB 3 – Completion of Sonar Car

- **Purpose:** The purpose of this lab is to have the student complete the robotic sonar car project.

- **Apparatus and Materials:**

    - See parts list at: http://www.cornerstonerobotics.org/excel%20doc/robotics_2_parts_list.pdf

- **Procedure:**
    - Complete the mechanical and electronic systems on the car. For the schematics see:
        - Sonar Car Circuitry 1: http://www.cornerstonerobotics.org/schematics/pic_programming_sonar_car1.pdf
        - Sonar Car Circuitry 2: http://www.cornerstonerobotics.org/schematics/pic_programming_sonar_car2.pdf
    - Program the car with sonar_car1:
        - Format in .pbp: http://www.cornerstonerobotics.org/code/sonar_car1.pbp
        - Format in .pdf: http://www.cornerstonerobotics.org/code/sonar_car1.pdf
- **Photos:**
    - To see past project solutions see: http://www.cornerstonerobotics.org/sonar_car_2008.php