**Lesson 46: IR Remote Control**

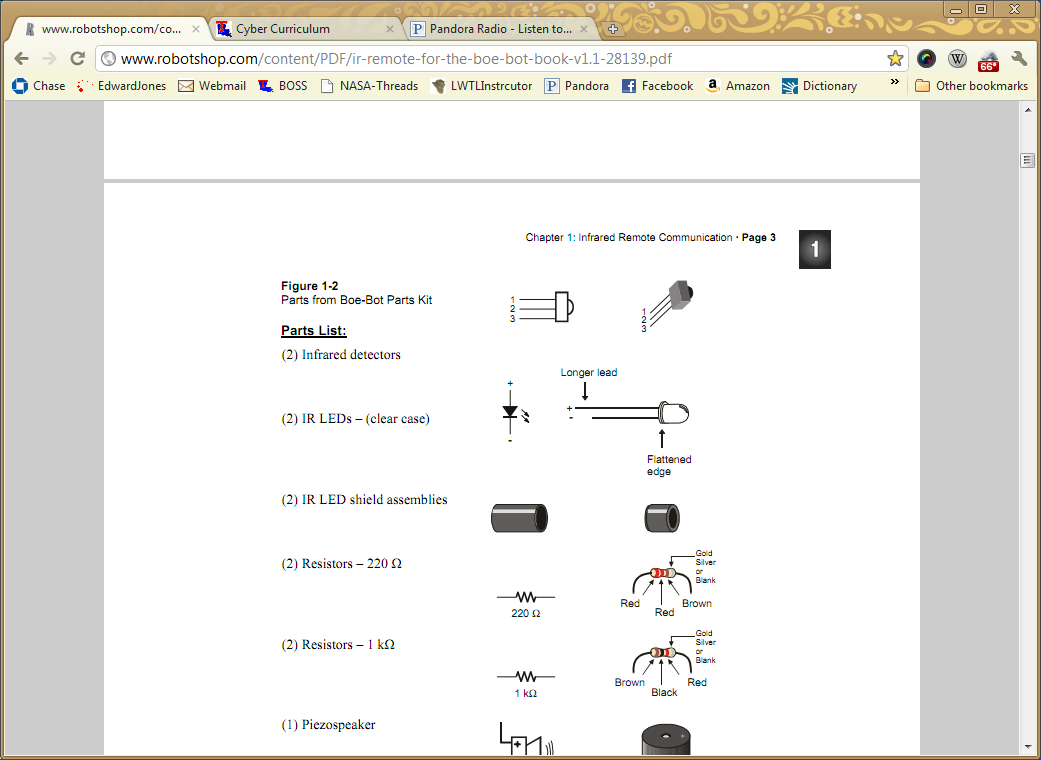
Needed

* Boe-Bot
* Computer with BASIC Stamp Editor program
* USB cable
* 220Ω resistor
* Jumper wires
* IR Detector
* Universal Remote

Infrared Remote Control

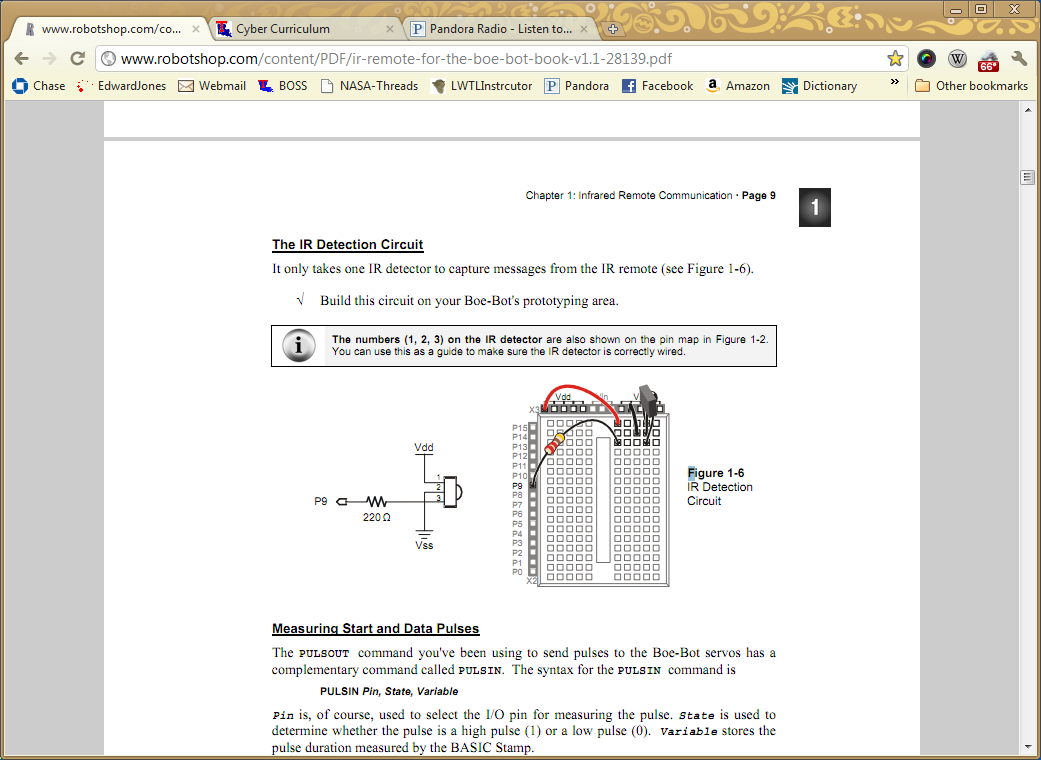
In this lesson, we will learn to control the Boe-Bot using an infrared detector in conjunction with a universal remote. It necessary to note here, that if you have not already done so, program your universal remote to work with a SONY TV. IR detectors for the Boe-Bot are made such that they respond to the SONY tv codes from a universal remote.

Now let’s discuss the IR detector and the circuit associated with it. The IR detector is shown in the following figure.



The IR detector senses the infrared output. The microcontroller can be programmed to use that information to perform a various task. For this lesson the IR detector will sense the IR output from a universal remote control and the microcontroller will use the information sensed from the IR detector to move the Boe-Bot servos.

The circuit for the IR detector in the Boe-Bot breadboard is shown below. Note: it is very important that the IR detector is connected properly (notice which side is facing to the right or left). If the IR detector is out into the breadboard backwards, it will get very hot.



Once the IR detector circuit is built, we should check out the various time measurements of the remote control buttons by using the following program. Record the values for each button in the given table. You will use the recorded times in future programming with the IR detector and the remote control, for instance navigation with the remote control.

' IR Remote for the Boe-Bot - RecordAndDisplayPwm.bs2

' Measure all data pulses from SONY IR remote set to control a TV.

' {$STAMP BS2}

' {$PBASIC 2.5}

time VAR Word(12) ' SONY TV remote variables.

index VAR Nib

' Display heading.

DEBUG "time ARRAY", CR,

"PWM MEASUREMENTS", CR,

"Element Duration, 2-us", CR,

"-------------- -------"

DO ' Beginning of main loop.

DO ' Wait for rest between messages.

PULSIN 9, 1, time(0)

LOOP UNTIL time(0) > 1000

PULSIN 9, 0, time(0) ' Measure/store data pulses.

PULSIN 9, 0, time(1)

FOR index = 0 TO 11 ' Display 12 pulse measurements.

DEBUG CRSRXY, 0, 4 + index, "time(", DEC index, ")",

CRSRXY, 9, 4 + index, DEC time(index), CLREOL

NEXT

LOOP ' Repeat main loop.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Remote Key | | | | | | | | | |
| Array Elem | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| time(0) |  |  |  |  |  |  |  |  |  |  |
| time(1) |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Array Elem | VOL- | VOL+ | CH- | CH+ | Enter | Power |
| time(0) |  |  |  |  |  |  |
| time(1) |  |  |  |  |  |  |

Infrared Navigation with Remote Control

Now let’s make this Boe-Bot move using the remote control.

Teachers this program will be difficult for the students to develop on their own. Discuss with them some guiding questions that will lead them to the program.

What is our objective? We want to program the Boe-Bot to move given different button inputs on the remote control

What information do we have about the different button inputs? We know the time associated with each button.

Is there a pattern that developed for certain buttons and their associated time durations? Yes, the time durations can be divided by using 500 as a distinction. They are either greater than or less than 500. There are four cases that occur:

1. Both time(0) and time(1) are < 500
2. Both time(0) and time(1) are > 500
3. Time(0) is < 500 and time(1) > 500
4. Time(0) is > 500 and time(1) < 500

We can use this to our advantage in order to perform four key navigation operations with the Boe-Bot: forward, reverse, right turn, and left turn.

The example program can be found on the next page. Once the students get the hang of this program you can challenge them to put the Boe-Bot in the maze a navigate through it using the remote. Also challenge the students to put a piezospeaker to make a beeping noise before the Boe-Bot backs up much like a school bus (NOTE: the Boe-Bot cannot perform two functions at once ie beeping and backing up at the same time. It would have to beep then back up). Challenge the students to refine the program make the buttons do different things (HINT: find out time durations for time(2), time(3), etc. this can be found by adding PULSIN 9, 0, time(2), PULSIN 9, 0, time(3), etc in the program that they wrote down time durations for. Using that information more patterns might form such that the students can have a fast forward and reverse, slow forward and reverse, fast turns, slow turns, and so on.)

' IR Remote for the Boe-Bot - 2BitRemoteBoeBot.bs2

' Control your Boe-Bot with an IR remote set to control a SONY TV

' with the 1-4 or CH+/- and VOL+/- keys.

'{$STAMP BS2}

'{$PBASIC 2.5}

time VAR Word(2) ' SONY TV remote variables.

DEBUG "Press and hold a digit key (1-4) or CH+/- and VOL+/-..."

DO ' Beginning of main loop.

DO ' Wait for rest between messages.

PULSIN 9, 1, time(0)

LOOP UNTIL time(0) > 1000

PULSIN 9, 0, time(0) ' Measure/store data pulses.

PULSIN 9, 0, time(1)

' Decide which maneuver to execute depending on the combination

' of pulse durations stored in the first two pulse measurements.

IF (time(1) < 500) AND (time(0) < 500) THEN

PULSOUT 13, 850 ' Forward

PULSOUT 12, 650

ELSEIF (time(1) < 500) AND (time(0) > 500) THEN

PULSOUT 13, 650 ' Backward

PULSOUT 12, 850

ELSEIF (time(1) > 500) AND (time(0) < 500) THEN

PULSOUT 13, 850 ' Right rotate

PULSOUT 12, 850

ELSEIF (time(1) > 500) AND (time(0) > 500) THEN

PULSOUT 13, 650 ' Left rotate

PULSOUT 12, 650

ENDIF

LOOP ' Repeat main loop.