**Lesson 51: Keypad I**

Needed

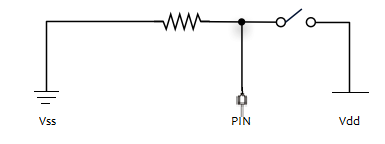
* Boe-Bot
* Computer with BASIC Stamp Editor program
* USB cable
* 10kΩ resistors
* Jumper wires
* Push buttons
* LEDs

Keypad

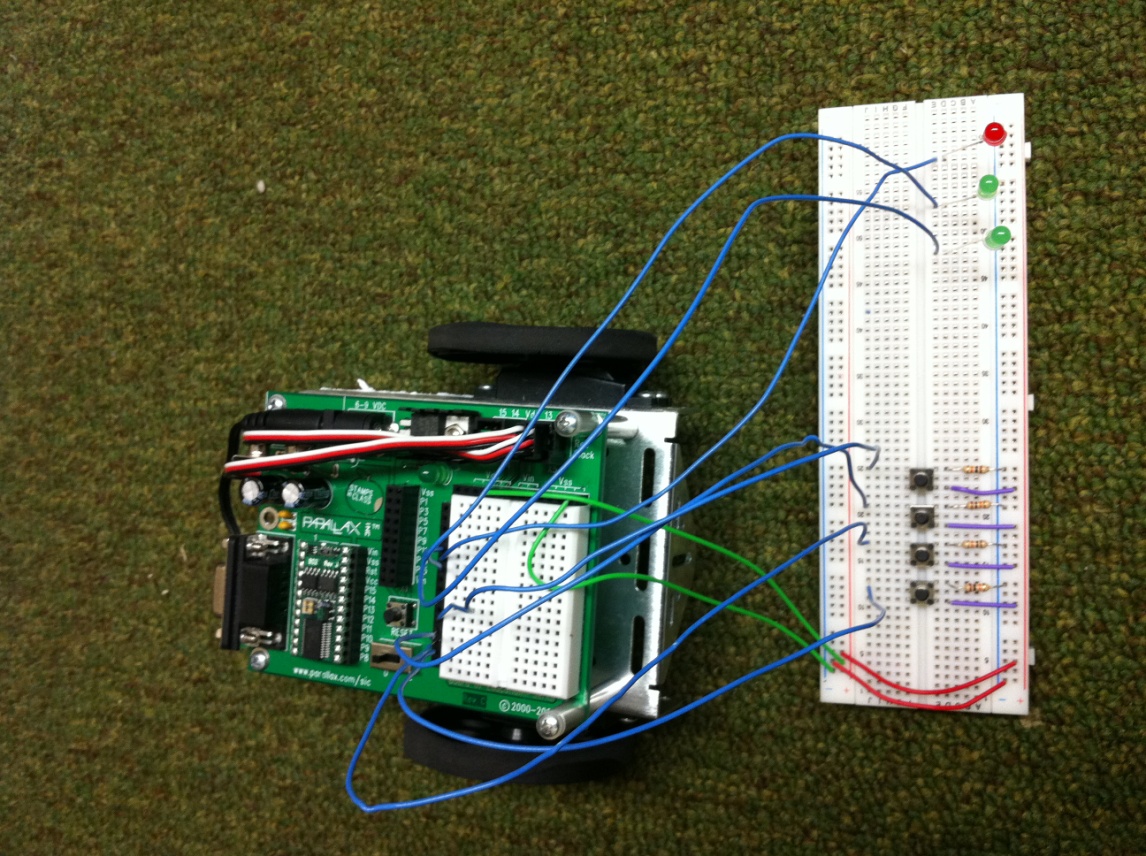
This lesson is a two day project, it will cover today’s Lesson and Lesson 55. To begin, the instructor will create a keypad on the Boe-Bot (depending on class size, the instructor probably should have a few Boe-Bots set up with different codes). During this first part of the lesson, the students’ objective is to use brute force to break the codes on each of the keypads. The second part of the lesson is for the students to create a keypad of their own and then write a program that uses another Boe-Bot to break the code, but more details about that later.

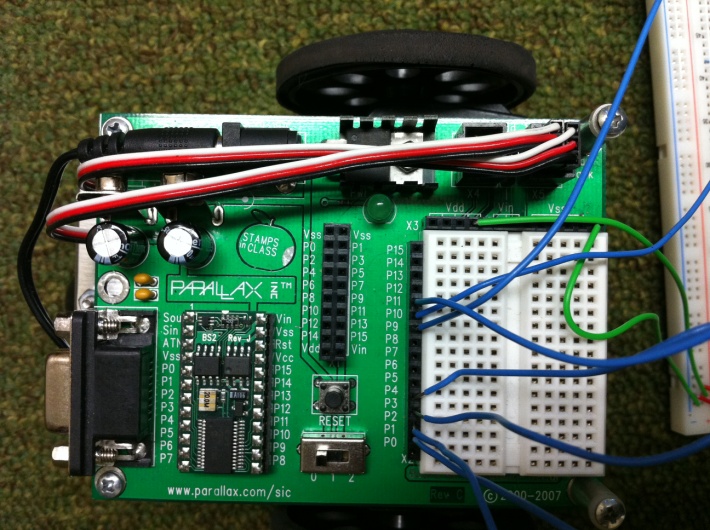
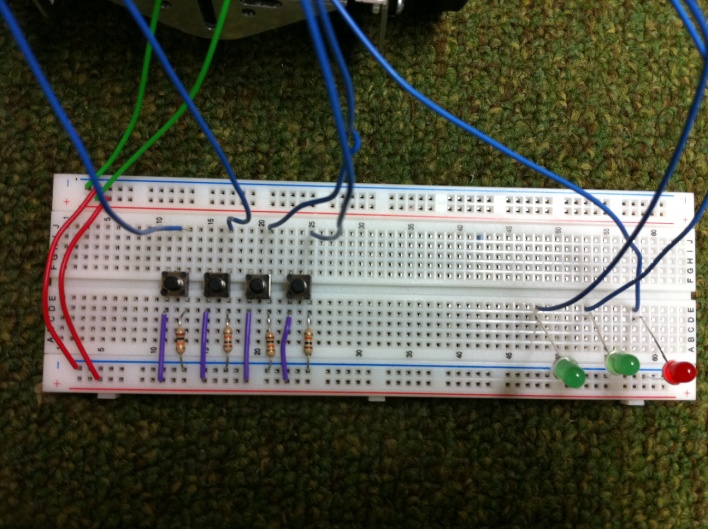
Creating the keypad

In order to create the keypad, you will use push buttons (aka tact switches), 10kΩ resistors (one for each push button), one red LED, two green LEDs, and some jumper wire. The circuitry for the push buttons is as follows.



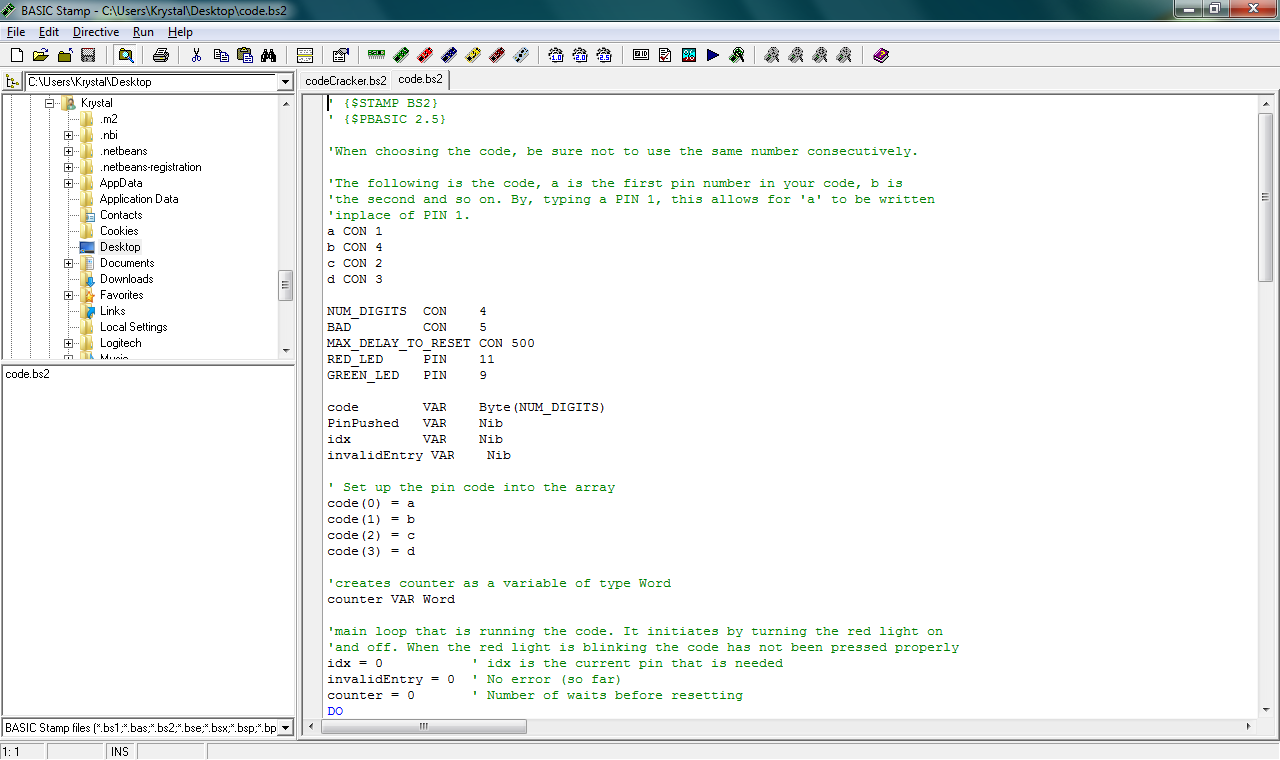
This above picture is for one push button, you can put as many push buttons as desired. The example used in the notes has four push buttons. In addition to the, the push button circuitry, three LEDs used to help identify if the code is pressed correct are also installed. The example uses one red LED (indicating incorrect button), and one green LEDs (the picture shows two green LEDs, we have since eliminated the additional green LED). Obviously the colors are arbitrary and you can use any colors you would like. Pictures of the completed circuit are shown below. For space sake, the example is shown on a protoboard connected to the Boe-Bot. This project can be done entirely on the breadboard of the Boe-Bot. You do not need protoboards, but they do help in providing more room for the circuitry.

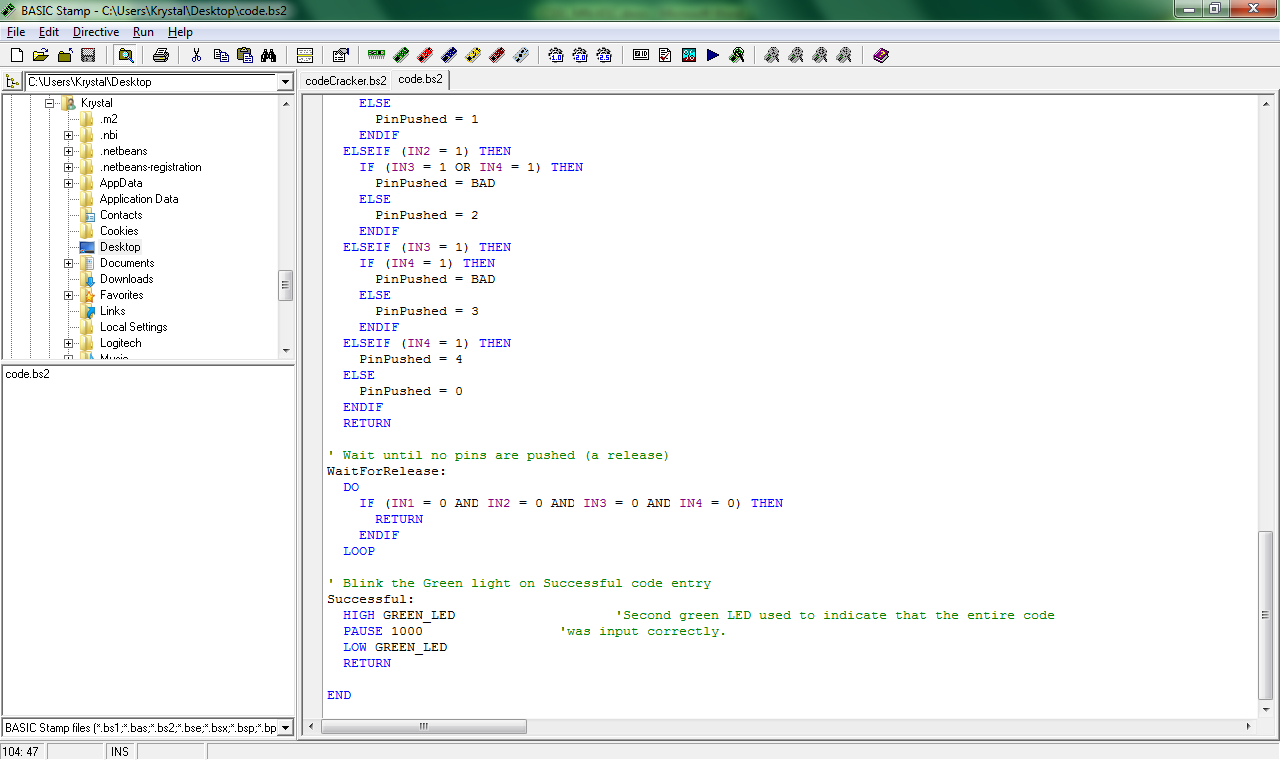
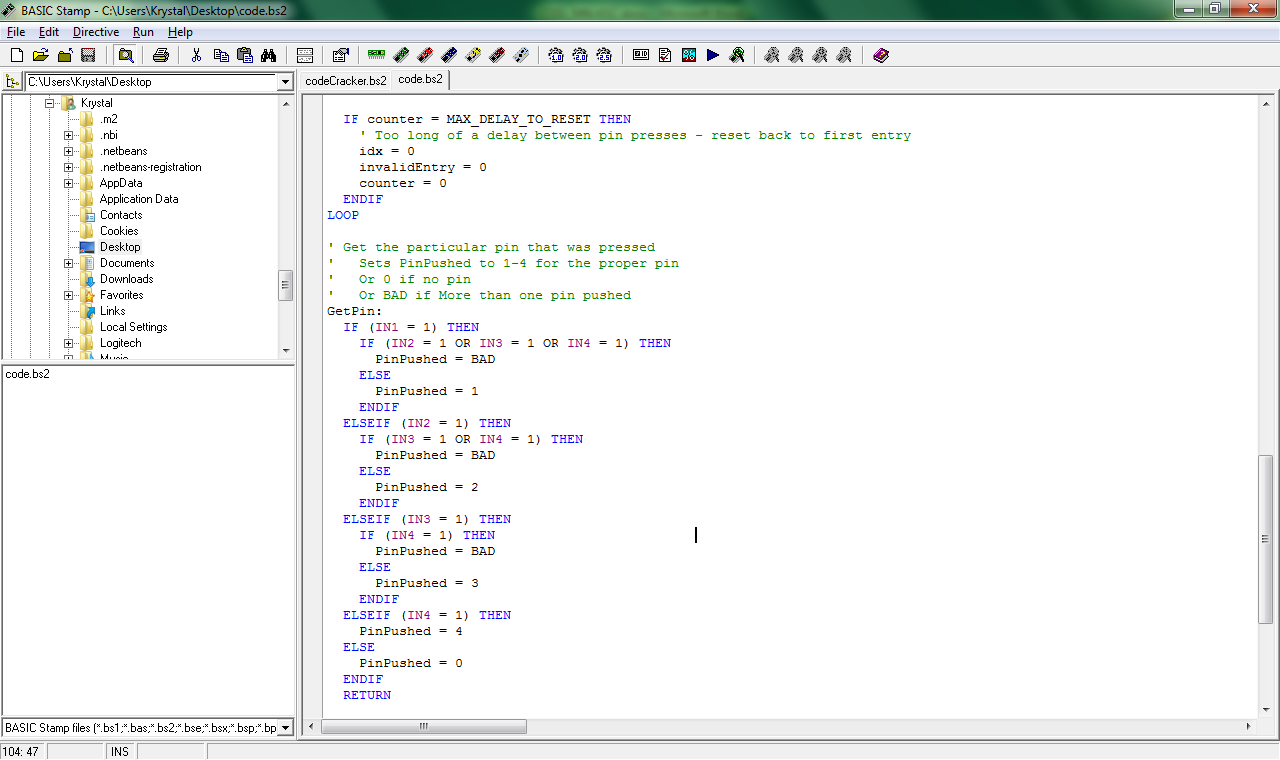
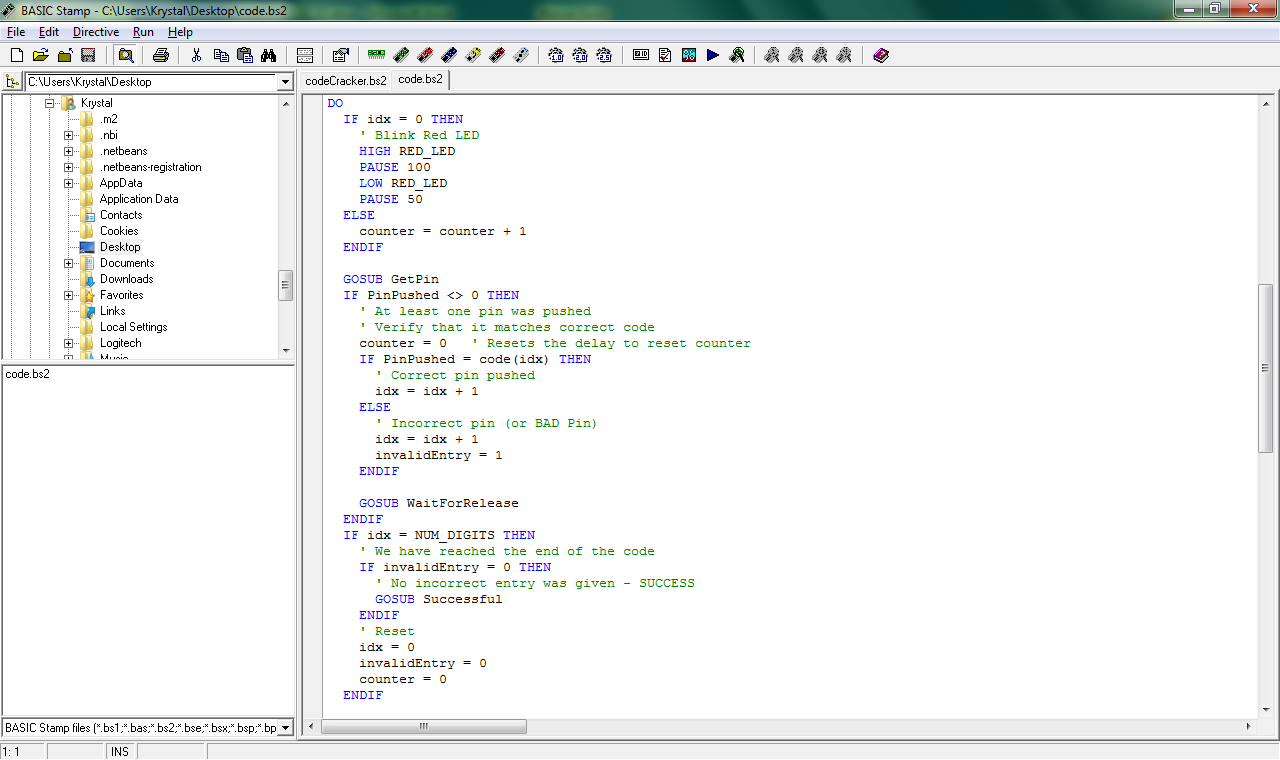




Writing the Program

The following is a sample program for the instructors to see only, in order to create their keypad codes (included are screen shots of the PBASIC program, as well as at the end is a text box of the program so that the instructor can just copy and paste the program into PBASIC). The programming code is well commented. There are a few things to note. First, to clarify, the code that you put in is associated with the pin number that the push button is attached to, if 2 is the first number in the code, then the push button attached to pin 2 is the correct first button. Also, the red flashing light indicates that the program is waiting for the user to push four buttons. The user must push four buttons. If the four buttons pushed were in the correct order a green light will flash. However if the four buttons pushed are in the wrong order, the red button will start flashing again and the program has been reset. It will now be waiting for a new set of four buttons to be pushed. After the students complete the task of breaking the codes, have them begin building the circuits on their Boe-Bots. You can even give them a copy of the code and let them analyze it (maybe make a flowchart from what they see in the code). The keypad code is a little complex, so depending on the level of programming comfort the students have teachers could either give them the code in its entirety and just walk through it with the student, give them the code with some missing pieces, or have the students attempt to create their own code. In Lesson 55, they will use the keypad code loaded on their Boe-Bot, but also create a different code that uses another Boe-Bot connected to the Bot that has the keypad circuitry on it. This additional Boe-Bot will be programmed to break the keypad code of the first Boe-Bot.





' {$STAMP BS2}

' {$PBASIC 2.5}

'When choosing the code, be sure not to use the same number consecutively.

'The following is the code, a is the first pin number in your code, b is

'the second and so on. By, typing a PIN 1, this allows for 'a' to be written

'inplace of PIN 1.

a CON 1

b CON 4

c CON 2

d CON 3

NUM\_DIGITS CON 4

BAD CON 5

MAX\_DELAY\_TO\_RESET CON 500

RED\_LED PIN 11

GREEN\_LED PIN 9

code VAR Byte(NUM\_DIGITS)

PinPushed VAR Nib

idx VAR Nib

invalidEntry VAR Nib

' Set up the pin code into the array

code(0) = a

code(1) = b

code(2) = c

code(3) = d

'creates counter as a variable of type Word

counter VAR Word

'main loop that is running the code. It initiates by turning the red light on

'and off. When the red light is blinking the code has not been pressed properly

idx = 0 ' idx is the current pin that is needed

invalidEntry = 0 ' No error (so far)

counter = 0 ' Number of waits before resetting

DO

IF idx = 0 THEN

' Blink Red LED

HIGH RED\_LED

PAUSE 100

LOW RED\_LED

PAUSE 50

ELSE

counter = counter + 1

ENDIF

GOSUB GetPin

IF PinPushed <> 0 THEN

' At least one pin was pushed

' Verify that it matches correct code

counter = 0 ' Resets the delay to reset counter

IF PinPushed = code(idx) THEN

' Correct pin pushed

idx = idx + 1

ELSE

' Incorrect pin (or BAD Pin)

idx = idx + 1

invalidEntry = 1

ENDIF

GOSUB WaitForRelease

ENDIF

IF idx = NUM\_DIGITS THEN

' We have reached the end of the code

IF invalidEntry = 0 THEN

' No incorrect entry was given - SUCCESS

GOSUB Successful

ENDIF

' Reset

idx = 0

invalidEntry = 0

counter = 0

ENDIF

IF counter = MAX\_DELAY\_TO\_RESET THEN

' Too long of a delay between pin presses - reset back to first entry

idx = 0

invalidEntry = 0

counter = 0

ENDIF

LOOP

' Get the particular pin that was pressed

' Sets PinPushed to 1-4 for the proper pin

' Or 0 if no pin

' Or BAD if More than one pin pushed

GetPin:

IF (IN1 = 1) THEN

IF (IN2 = 1 OR IN3 = 1 OR IN4 = 1) THEN

PinPushed = BAD

ELSE

PinPushed = 1

ENDIF

ELSEIF (IN2 = 1) THEN

IF (IN3 = 1 OR IN4 = 1) THEN

PinPushed = BAD

ELSE

PinPushed = 2

ENDIF

ELSEIF (IN3 = 1) THEN

IF (IN4 = 1) THEN

PinPushed = BAD

ELSE

PinPushed = 3

ENDIF

ELSEIF (IN4 = 1) THEN

PinPushed = 4

ELSE

PinPushed = 0

ENDIF

RETURN

' Wait until no pins are pushed (a release)

WaitForRelease:

DO

IF (IN1 = 0 AND IN2 = 0 AND IN3 = 0 AND IN4 = 0) THEN

RETURN

ENDIF

LOOP

' Blink the Green light on Successful code entry

Successful:

HIGH GREEN\_LED 'Second green LED used to indicate that the entire code

PAUSE 1000 'was input correctly.

LOW GREEN\_LED

RETURN

END