Brenda Boudaie

EE / CS 120B

March 17, 2019

Custom Lab Project: Bop It

INTRODUCTION

Bop It is an interactive game where the player follows a series of commands issued through the toy. The game has multiple inputs including buttons, pull handles, twisting cranks, wheels, flickable switches - with pace speeding up as the player progresses.

In my implementation of this game, I supply the player with three interactions: FlickIt, WaveIt, and of course BopIt. The user guide and specifications of the technology can be found bellow.

USER GUIDE

When the game is turned on, the player can begin by pressing BopIt. Once BopIt is pressed, one of three LED's will randomly light bellow a trigger, indicating which command to hit. The three commands can be triggered by the following:

FlickIt — flick the joystick in any direction, pressing down will not reward a correct move.

BopIt — press down on the touchpad.

WaveIt — wave your hand above the motion sensor. To ensure the motion is getting detected, wave closer to the top of the sensor.

As the player consecutively gets the correct command, the accepted answer time decreases down from 1 second.

There are two ways to lose the game. One, is to answer incorrectly (ie: FlickIt instead of BopIt). Another, is to miss the command within the time frame.

GameOver, is denoted by the very bright red LED light at the top of the board (resistor intentionally not added to amplify that the player has lost). This light will remain on for 5 seconds, then turn off. The game can then be started again by hitting BopIt.

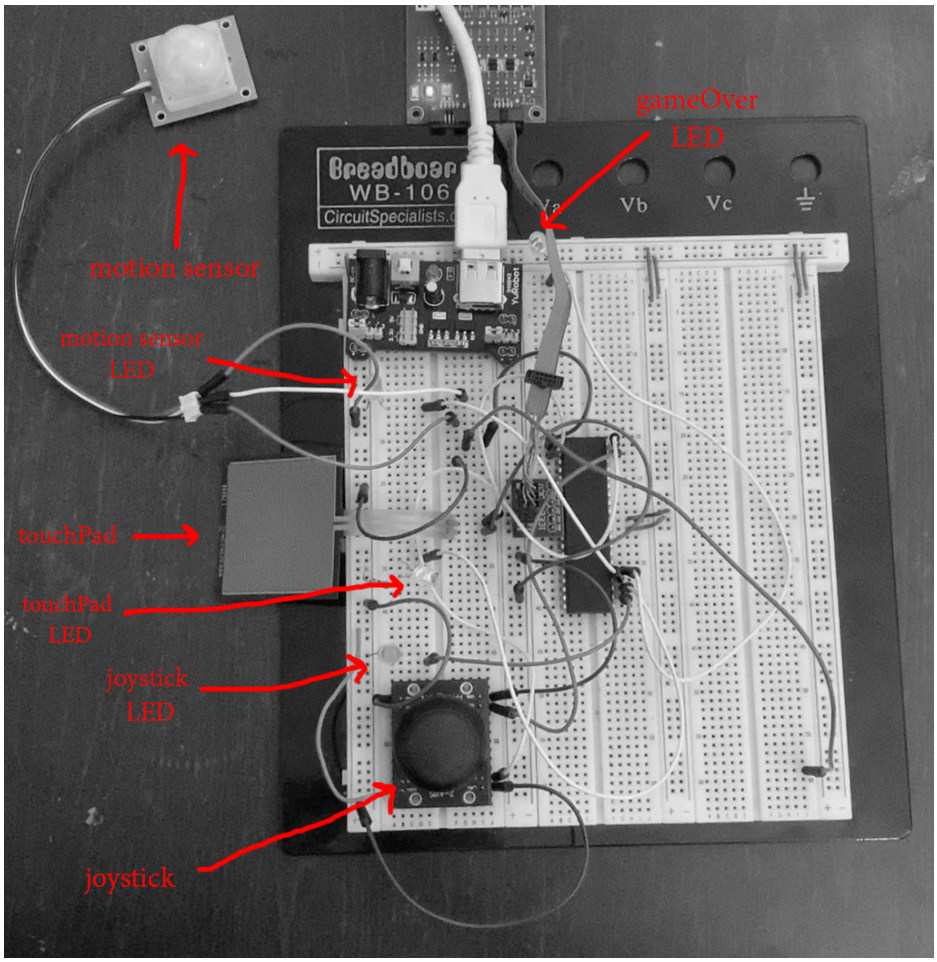
Tech Specs

The software was written through Atmel Studio, in C. The program is essentially running from two state machines: GameStates and InputStates.

GameStates holds all of the game logic. Meaning, it instructs the board to randomly display one LED at a time, switch to another LED only when the correct input is given, decrement the timer as the game progresses, and turn on the GameOver light once the player triggers the wrong input. The LED's light randomly just by standard library's rand() function. This function is placed within a switch statement, where each case (0, 1, 2) correlate to a certain PORT. The timer is kept through a counter, which decrements every time the correct in put is given. Similarly, the gameOver LED is given a counter that decrements 5 seconds, based off of the given period.

InputStates deals with making sure the inputs are read correctly. The two states "Input Start" and "Input Down" ensure that the board does not double read an input, because of debouncing. For example, if a player flicks the joystick but holds it for a little in its position, InputStates makes sure that the board does not read that as two back to back inputs.

The hardware is using the ATmega1284 AVR microchip. The structure of the board contains the microchip in the middle, the gameOver LED on the upper left hand side, the three complexities on the bottom of the board, with their corresponding LED's above them. An image of the set up can be found bellow as well.



COMPONENTS

Inputs

- spark fun joystick
- touchpad
- motion sensor

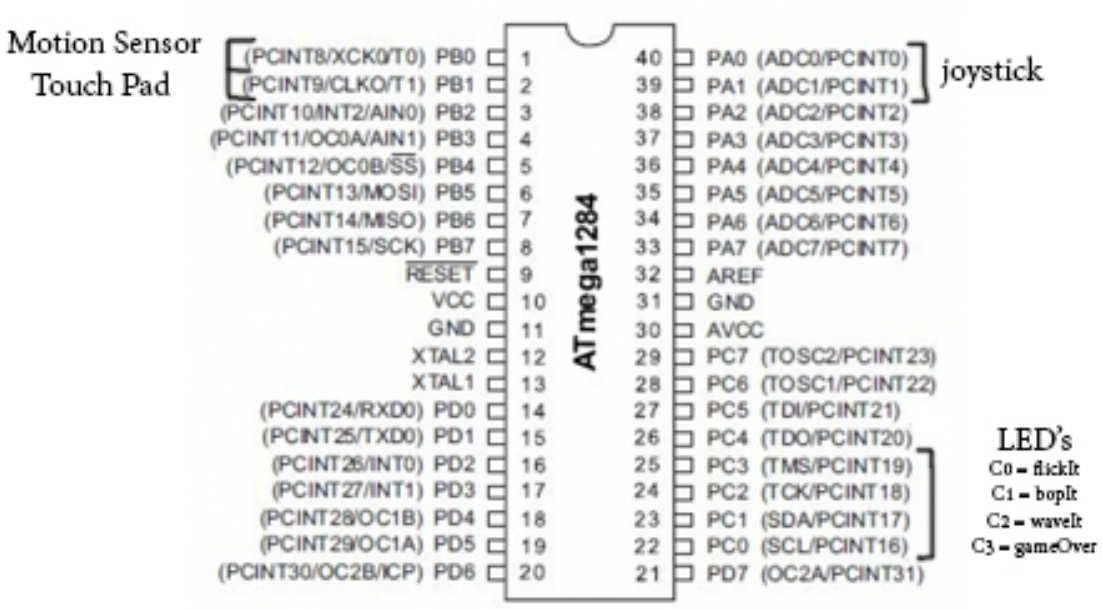
Outputs

- 4 Lab Kit LED's
  - One for GameOver state, three for the corresponding inputs

COMPLEXITIES / BUILD-UPONS

- touch pad (1)
- sparkfun joystick (1)
- motion sensor (1)

PINOUT



OTHER REMARKS

There are a couple of small issues with the game to be aware of:

(1) The motion sensor is very sensitive. In order to get the game to actually work, the player needs to keep the sensor stable for about 10 or 15 seconds. This makes sure it is fully static, so the sensor does not falsely read. Otherwise, the game automatically gives you a gameOver for inputting the wrong trigger.

(2) The set up of the board does not separate the complexities far enough from each other, so an issue I noticed while having friends test the game, was that their hand would accidentally press on the touchIt while trying to flickIt. So, an improvement would be to distinctly separate the two complexities enough that this problem does not occur, and give the player an automatic gameOver.

Other than those two, I, as well as a handful of my friends, could not find other issues while testing the game. I hope you enjoy my final custom lab project of the game Bop It :)