Embedded Pinball Project

Matan Orland - 300514643, Ori Bar-El - 305135956 02/04/2018

A video of our finished product can be accessed here.

Introduction

Our project is a pinball game based on a CC-1350 launchpad that is connected to 2 servos that function as flippers and a force sensor that is used to encounter that the ball fall into the drain. The game is controlled through a dedicated Android and TI-RTOS applications that communicate over BLE. The Android app transmit the flippers' state to the board using the TI-RTOS application and gets notifications from the TI-RTOS application with regard to the number of lives left.

Hardware Setup

Hardware

1. Two servo motors - We used TowerPro SG90 motors.
2. A pressure sensor - We used the FlexiForce 1lb pressure sensor.
3. A $470\mu F$ capacitor.
4. a $10k\Omega$ resistor.
5. Breadboard.
6. Wires.
7. A marble.
8. Some cardboard.

Circuit

Below is a diagram of our circuit:

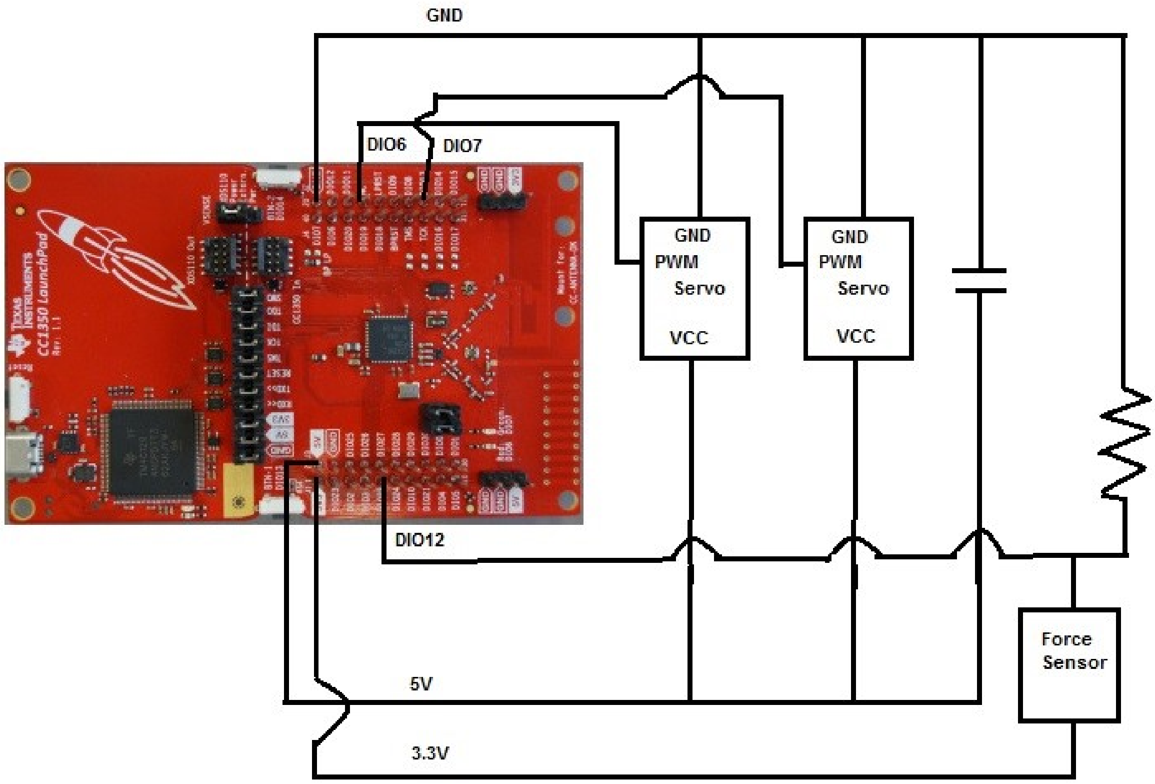

Figure 1: Circuit description - For clarity DIO pins are spread out, and do not match their actual locations on the board.

Software Setup

Pre-requisites

1. Java 8 (or above)
2. Android Studio
3. Android SDK 27
4. Android Build tools v27.0.2
5. Code Composer Studio

Installation Steps

1. Clone the project from our repository.
2. Connect your Android device to the computer and enable the developer's debug mode in your Android device.

3. Download the app to your android device by running the application in Android studio.
4. Connect the TI CC-1350 to your computer using the dedicated cable.
5. Run simple_peripheral_cc1350lp_stack_FlashROM (this step should be done only once).
6. Run simple_peripheral_cc1350lp_app_FlashROM and start controlling the launchpad using the Android app.

Mobile application

The mobile application was developed in Java. Figures 2-4 show the user interface. The application implements the following functionality:

1. Scanning for BLE devices.
2. **Automatic** pairing with the CC-1350.
3. Support for newest version of Android (gaining permissions for both Bluetooth enabling and access to the device's location).
4. A "start new game" button that resets a timer in another thread
5. Two buttons for controlling the hardware flippers using a write action. The buttons also contain a continuous press handler to support a continuous lift of the flippers.
6. A notification handler that gets notifications over BLE from the board regarding the number of lives left.

Main Challenges, or "How I learned to stop worrying and love the servo"

1. Using the BLE protocol for both writing and notifying.
2. Writing an Android application - We had to learn some basic android application design, UI of an android application and using BLE in an android application.
3. Wiring and control of the servos using the built-in PWM pins.
4. Noise reduction when more than one servo is connected - This is a problem we encountered because we powered the servo motors directly from the board. When one motor was connected, no problem was noticeable, but when two motors were connected a very noticeable noise was felt. This was due to the nature of servo motors, requiring constant power. A slight change in PWM frequency meant a lot of noise. We addressed this problem by add a $470\mu F$ capacitor for decoupling purposes. This worked surprisingly well (using a smaller capacitor than the one we used did not facilitate the problem).

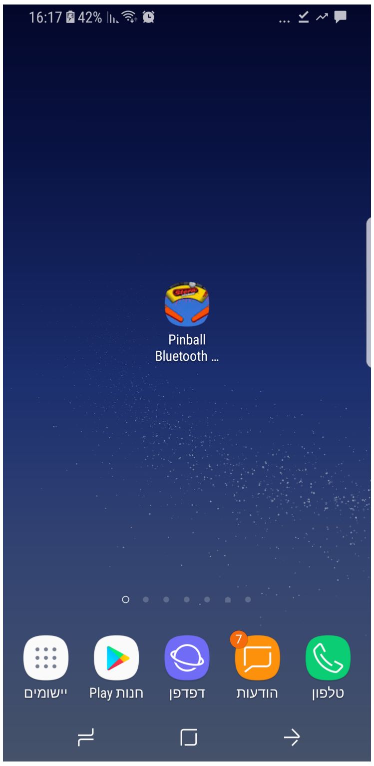

Figure 2: Application icon

5. Using a sensor to measure strikes - This was done using a force sensor, however we feel as though this is not the ideal solution, as a small ball (a marble in our case) will only exude very small force for a very short time. A better solution would have been to use a metal ball and a Hall effect sensor.

6. Due to the nature of a pinball game a timely response to a button press is crucial. This fact required controlling the packets' transmission rate to allow for high responsivity and to achieve a real-time game experience. To handle this we extensively explored the TI code to detect the relevant configuration settings and a fiddled with them to get the required behavior.

7. Handling continuous and momentary button press in the application - This is something that, logics-wise is actually easier when controlling the game from the Android application. We had a message written to the board on button press, and on button release. This meant that addressing button press/release could be done in a HW interrupt handler. A solution for HW controlled buttons would have been to launch a thread on button press which constantly checks whether the button is still pressed. This is a lot more resource inefficient. After dealing with responsivity issues we arrived at a solution which meant that using the app felt very natural.

Figure 3: Granting Bluetooth permissions

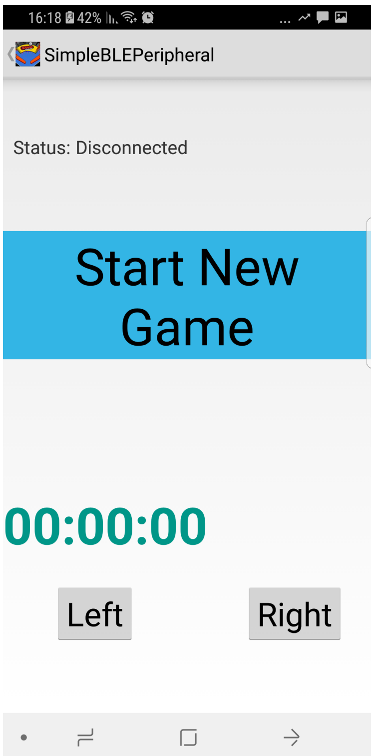

Figure 4: Application main page