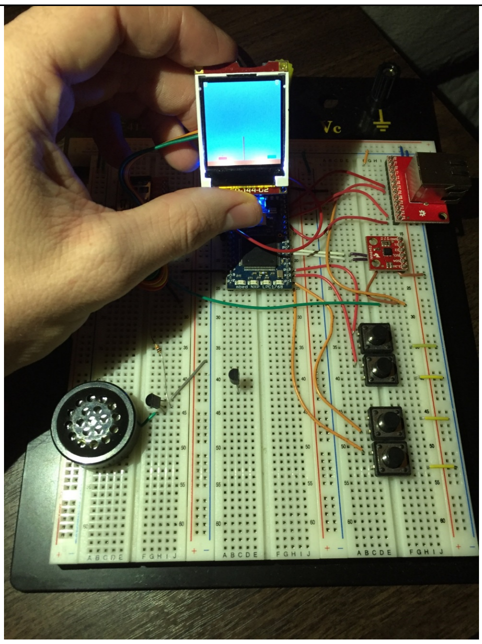
ECE 2035 **Project 2: Badminton Game** Summer 2016

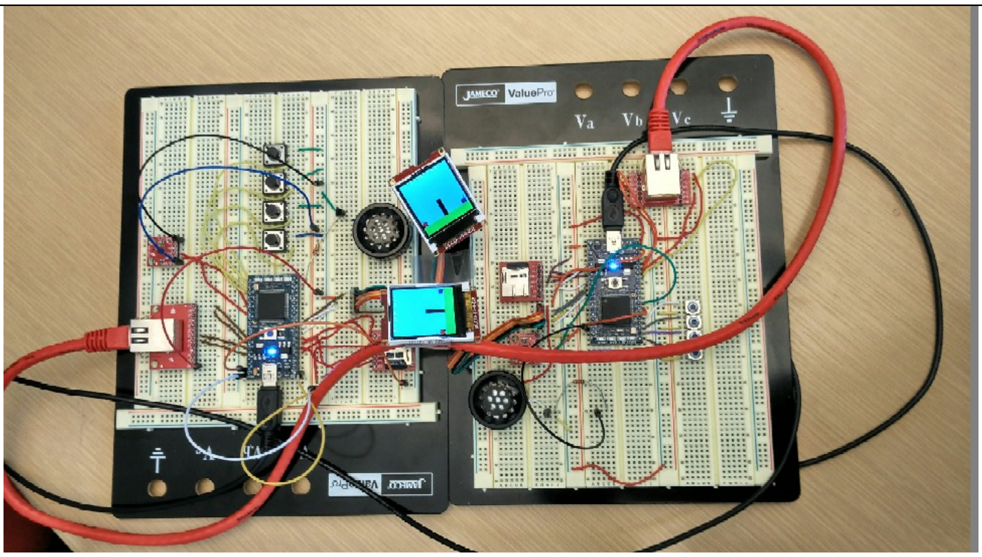
This project creates an Mbed version of an interactive Badminton game. In this game, players strike a shuttlecock (shuttle for short) with a racket to send it back over the net. To score a point, the shuttle must land in the opponent's court, or the opponent fails to return the shuttle to the player side. In this game the position of each player's racket is controlled by the accelerometer position. In the baseline game, the racket can move in two dimensions, and when struck the shuttle always moves with a constant initial velocity toward the opposing player. The baseline game may be played in single player mode in which the other player is controlled by the game, and in two player mode, where the second player's mbed setup is remotely controlled over an Ethernet link.

To use the Ethernet link you have been provided with an Ethernet API called the Game Synchronizer. It allows you to connect your mbed to a second "slave" mbed over Ethernet and to draw basic graphics primitives to either or both screens.

There are a multitude of extra credit options available to you on this project (to be detailed later in the assignment description), and we recommend you start early so that you are able to take full advantage of them.



Example Badminton screen



A pair of Mbeds connected by Ethernet

Game Rules

- Each player is able to see an identical view of the game court.
- Players alternate shots.
- Each shot must be returned unless it goes out of bounds.
- Any failed return (excluding out of bounds) results in a point to the opposite player.
- The serve belongs to the player who scored the previous point

Implementation

In this project, you will be given shell code that implements parts of the game, and you will complete the rest. You must complete the basic game functionality (described below) for 75 points. You can earn additional points by successfully implementing and documenting extra features to enhance the basic game (as described below). These are worth 5 points per extra feature, for a maximum of 50 additional points (10 additional features). So, to get a 100% on this project, your mbed program must support the basic game and 5 additional features.

Basic Game Functionality

The following features, denoted in the shell code by a "TODO" comment, are baseline features to be completed. These features must be implemented, documented and working correctly to earn the baseline 75 points:

Game Entry Menu: At startup, a menu screen should appear on player 1's uLCD asking the player whether they wish to play in single- or multi-player mode. This will require access to player one's uLCD and push buttons on the mbed board. (The menu in the shell code always defaults to SINGLE_PLAYER mode.) Additional modes may be available depending on your particular implementation. For full credit, you must display reasonable effort to make the menu appealing to potential players. Be creative!

Supporting 1-Player/2-Player Game Control: Implement the game logic for the two modes of play (one or two player). In one player mode, the pushbuttons and accelerometer switch focus immediately after the shot is taken, and then are used to control the other side.

In two-player mode, the Student side mbed program is the main program in charge. It computes what to do with the inputs and how to update both screens. The TA side mbed is "slaved" to the Student mbed. It does nothing but gather accelerometer and pushbutton inputs and send them to the Student side for processing. The student side sends back screen display commands that the TA side follows to update its screen. The TA side code is provided in an executable named **`TA.bin`**. To play your game in 2-player mode with another student, load one mbed with the executable for your game and load the other student's mbed with the TA side executable. Connect the two boards together using your Ethernet cable and reset both mbeds to play!

(A demo version student side program **`Student.bin`** is provided on T-square, along with **`TA.bin`** so you can get an initial idea of the game play.)

Shuttlecock movement: The shuttle should follow the physics of motion for an object launched with a given initial velocity (remember velocity is a vector).

$$y(t) = y_0 + v_y t - \frac{1}{2} g t^2$$
$$x(t) = x_0 + v_x t$$

IMPORTANT:
To ensure your x and y values are compatible with the discrete nature of the game world, you should use the floor() function from <math.h> to truncate your x and y values.

$$y(t) = floor\left(y_0 + v_y t - \frac{1}{2} g t^2\right)$$
$$x(t) = floor(x_0 + v_x t)$$

Landscape: Enhance the background landscape by adding features to it. Features may interact with the shuttle. An example might be an overhanging branch.

Score: Display a small indicator somewhere on the screen indicating the score.

Sound effects: Your game should play a fun sound at the beginning when the menu screen appears and a game over sound at the end of the game. Make an interesting sound when the shuttle is hit. Be creative!

Game Over: Your program needs to determine and display who won the game. Include some visual effect and text notifying the player that the game is over. As with the game menu, we encourage you to be creative! Text on a plain background probably won't cut it.

Extra Features

ONCE ALL OF THE BASELINE FEATURES HAVE BEEN IMPLEMENTED, you can implement a subset of these features to earn an additional 5 points per feature, up to a maximum of 50 points:

Shot angle: Use a pushbutton pair to control the angle of the racket. Set the direction of the shuttle initial velocity vector to be normal to the racket

Shot strength: Use a pushbutton pair to control the magnitude of the shuttle initial velocity.

Use pushbuttons to create a new feature or powerup that can be used for a *limited total time*. Think outside the box! Examples include the following:

> **Velocity power up:** Create a special power to that allows a player to move faster for a fixed amount of time after the press of a button.
>
> **Shuttle damping field:** Create a special feature that reduces the initial velocity of the opponent's return shot
>
> **Magic bird:** Causes the Foo Bird of happiness to magically appear and knock the shuttle to the ground in opponent's court.
>
> **Split Power Up:** Create a powerup that causes the shuttle to split into two, both of which must be returned by opponent.

Add new hardware to do something interesting, such as using the RGB LED to indicate some status information with color (http://developer.mbed.org/users/4180_1/notebook/rgb-leds/).
You can use different color of LED to indicate different situations, for example:

1. Turn the LED Red indicating that your opponent is at game point
2. Turn the LED Blue indicating you are at game point.

Difficulty Levels: for configuring the game (e.g. starting it at some level: Easy/Medium/Hard).

Keep track of game history and show in interesting way (e.g., Highscore, 2-player game best of 3 rounds) – this requires that you reset the game using a pushbutton without using the mbed's reset button which reinitializes the entire program or saving the state of the game history so that it can be reloaded when the mbed is reset.

Enhanced Graphics: Create a dramatic shuttle animation, or racket animation

Add sound effects – be creative. These could accompany advancing to a new level, gaining or losing a life, winning a match, etc. Since the code already uses the MBED real-time OS, you should create a thread for playsound() to minimize its interference with gameplay. As an alternative for short sounds or playing single notes, there is documentation on "Playing a Song with PWM using Timer Interrupts" here:http://developer.mbed.org/users/4180_1/notebook/using-a-speaker-for-audio-output/
Hello world song player code URL is http://developer.mbed.org/users/4180_1/code/song_demo_PWM/

Video of a media player: http://developer.mbed.org/users/4180_1/notebook/mpod---an-mbed-media-player/

Other Feature Ideas: If you have an idea for a feature along the lines of the ones listed above, ask a TA or Professor whether it will qualify. Our emails are available on the class site.

Shell Code
The shell code is available at:
https://developer.mbed.org/teams/ECE2035-Spring-2015-TA/code/2035_Badminton_Shell/
Click the “Import this program” button (upper right) to bring this project into your own account.

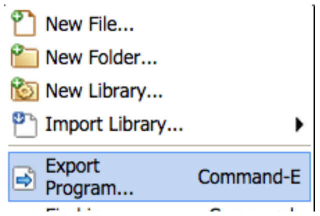
Where to begin?
We recommend starting by reading the shell code and comments, and the API document. Then start small, say create and move the shuttle. A key concept is the notion of a frame cycle. In each cycle or interval of time, the program must compute what will happen next for each object in the game based on the current game state and inputs, and then make those updates. You will need to “undraw” objects that have moved between frames before you redraw the objects in the next frame.

What is and is not allowed?
For this project, you are given free reign over how to implement your game; for all intents and purposes, anything goes (within the scope of the honor code, of course). The only requirements we have placed upon you are the baseline features that you should implement as well as any additional features that should you choose to implement. The shell code serves as a starting point and is not the only way to implement features. If you want to start fresh with a new project, then you may do so as long as you complete the required features.

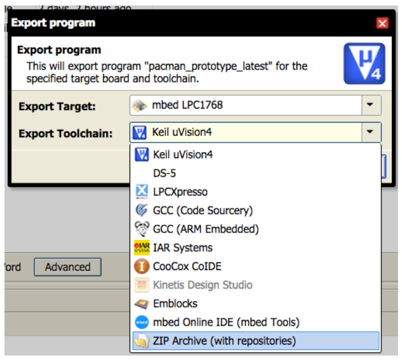
Checklist
Also, a feature checklist file named `P2-checklist.txt` is available on T-square. Put an X before each feature in the checklist that your program implements and that you will demonstrate to the TA.

Project Submission
In order for your solution to be properly received and graded, there are a few requirements.

1. Compile your project and name the executable `P2.bin`.
2. Create a `.zip` archive of your project using the following steps:
   - In the Program Workspace (the left hand vertical scrolling window in the mbed compiler), right-click the project name of your project and select "Export Program..."
   - A box will pop up. Choose "ZIP Archive" in the Export Toolchain menu:



- Select "Export" and another dialog box will pop up that will allow you to save the file. Name this archive **`P2.zip`**.
- Upload **`P2.zip`**, **`P2.bin`**, and **`P2-checklist.txt`** to T-square before the scheduled due date, **5:00 pm on Monday, 18 July 2016** (with a one hour grace period).



Export program

Export program

This will export program "pacman_prototype_latest" for the specified target board and toolchain.

Export Target: mbed LPC1768

Export Toolchain: Keil uVision4

3. After your project zip file is uploaded to T-square, meet with a TA and demo your game on **your** hardware. While the TA is watching, you will download the executable you submitted from T-square to your mbed, and then demonstrate all of the features of your game. Bring a printout of the checklist you submitted showing which features you successfully implemented so that the TA can confirm them. This must be completed by **Tuesday, 26 July 2016.**

You should design, implement, and test your own code. There are many, many ways to code this project, and many different possibilities for timing, difficulty, responsiveness and general feel of the game. Your project should represent your interpretation of how the game should feel and play. Any submitted project containing code (other than the provided framework code and mbed libraries) not fully created and debugged by the student constitutes academic misconduct.

Project Grading - The project weighting will be determined as follows:

description	*percent*
Basic program functionality	
Technique, style, comments	15
Baseline features	60
Advanced Features	50
total	125/100 maximum

Extra Extra-Credit: Here's another chance to earn extra credit (and the admiration of your fellow students). Create a short (< 1 minute) video clip highlighting the coolest feature(s) of your mbed Badminton project. If you enter a high quality video, you will earn an extra credit point on your overall course grade. (This is particularly helpful if you are on a letter grade boundary).
Your video clip must clearly identify the features that you are highlighting. To submit your entry, save your video clip as a **`.mov`** or **`.mp4`** file named "**`P2clip.mov`**" or "**`P2clip.mp4`**" and upload it by **Friday 29 July 2016** to T-square under Assignments > "Badminton Highlights".
We'll show a highlight reel of the video clips in lecture on the last day of classes with special recognition going to the top entries.

References

1. Shell code: https://developer.mbed.org/teams/ECE2035-Spring-2015-TA/code/2035_Badminton_Shell/
2. Mbed Pinout:
   http://mbed.org/nxp/lpc2368/quick-reference/

3. Mbed-NXP Pinout:
   http://mbed.org/users/synvox/notebook/lpc1768-pinout-with-labelled-mbed-pins/
4. NXP-LPC1768 User's Manual:
   http://www.nxp.com/documents/user_manual/UM10360.pdf