Virtual Softball

6.111 - 2017

Katie Shade and Melinda Szabo

1. Overview

Softball is a team sport played by people of all ages. It relies on both strategy and skill in order to be successful. In the winter months, it is often difficult to practice outside due to weather. This virtual softball game allows users to swing a real bat and practice their hitting timing from the comfort of the indoors. It also gives non-softball players the opportunity to experience a life-like version of the game with real equipment.

Users will stand with a bat in front of a screen. When the game begins, users will see a ball approaching them as it becomes larger on the screen. It will be located in one of the four quadrants of the strike zone. As the ball approaches the batter, this will be visually cued by the ball changing color. When the ball is within hitting distance of the batter, it will turn a bright color, indicating the optimal time to hit the ball. It will stay that color for a short period of time until it returns to its original darker color and stays frozen on the screen in the strike zone.

The user is expected to swing the bat when the ball is in a hittable position (a bright color). By interfacing with an IMU, real bat position and angle from an accelerometer can be recorded, and a virtual representation of the bat will be reflected onto the screen in front of the user after the swing. The goal is to give the user feedback on both their hitting timing as well as their batting mechanics in the form of the bat angle while swinging. The screen will indicate approximately where the bat hit the ball (up, down, left, right, center) or if they missed the ball due to timing or incorrect bat angle. If the ball is hit, a future plausible trajectory of the hit ball will be shown in order to make the game more fun and give better feedback to the user.

With time permitting, there will also be a defensive component to the game in the form of a field player virtually catching a ball. For this part, the field player would see a ball coming at them on the screen as the ball becomes larger when it approaches. They will then move a glove to catch the ball at the right time. The ball's location will be determined by the way the batter (described in the previous paragraph) hit the ball. The movement of the glove can be implemented either in a videogame-like manner with moving arrow keys or a joystick or via a physical glove that is tracked. These defensive components are considered stretch goals as the batting functionality is the first priority.

2. Design

2.1 Goals and Scope

Priorities are listed in terms of importance. **P0** denotes baseline requirements to build the rest of the project on. Anything **P3** or higher is definitely considered a stretch goal. The priorities list the general order in which the goals are expected to be worked on.

2.2.1 Baseline

1. Take inputs from physical bat swing
    - **P0 -** Determine the timing of when the user would hit the ball if this were a real game.
    - **P0 -** Determine the angle of the physical bat as it comes through the strike zone.

2. Graphically represent a ball's path as it comes toward the user.
    - **P0 -** Ball becomes larger as it gets closer to the batter.
    - **P0 -** Use color to represent when the ball should be hit.

3. Graphically represent how the bat was swung.
    - **P0 -** Show the location on the ball that the bat is expected to hit it.
    - **P0 -** Show if the batter missed the ball.

2.2.2 Expected

1. Ball location in the strike zone will change between pitches.
    - **P1 -** The ball position will change randomly from pitch to pitch.
    - **P1 -** The corresponding determination of a hit or miss will be dependent on both the bat swing and the ball position.

2. Indicate that the user is ready for the next pitch.
    - **P1 -** The user can press a button to show they would like to receive another pitch.

3. Wired bat communications with the NEXYS4.
    - **P1 -** The IMU on the bat will be sending data via a wired connection to the necessary system modules. Although having a wire connected to the bat is not ideal, the bat can still be swung normally such that the swing is not affected by the wire.

4. Graphically show batted ball trajectory.
    - **P2 -** Based on where the bat hit the ball, show a plausible trajectory for the ball after being hit.

2.2.3 Stretch

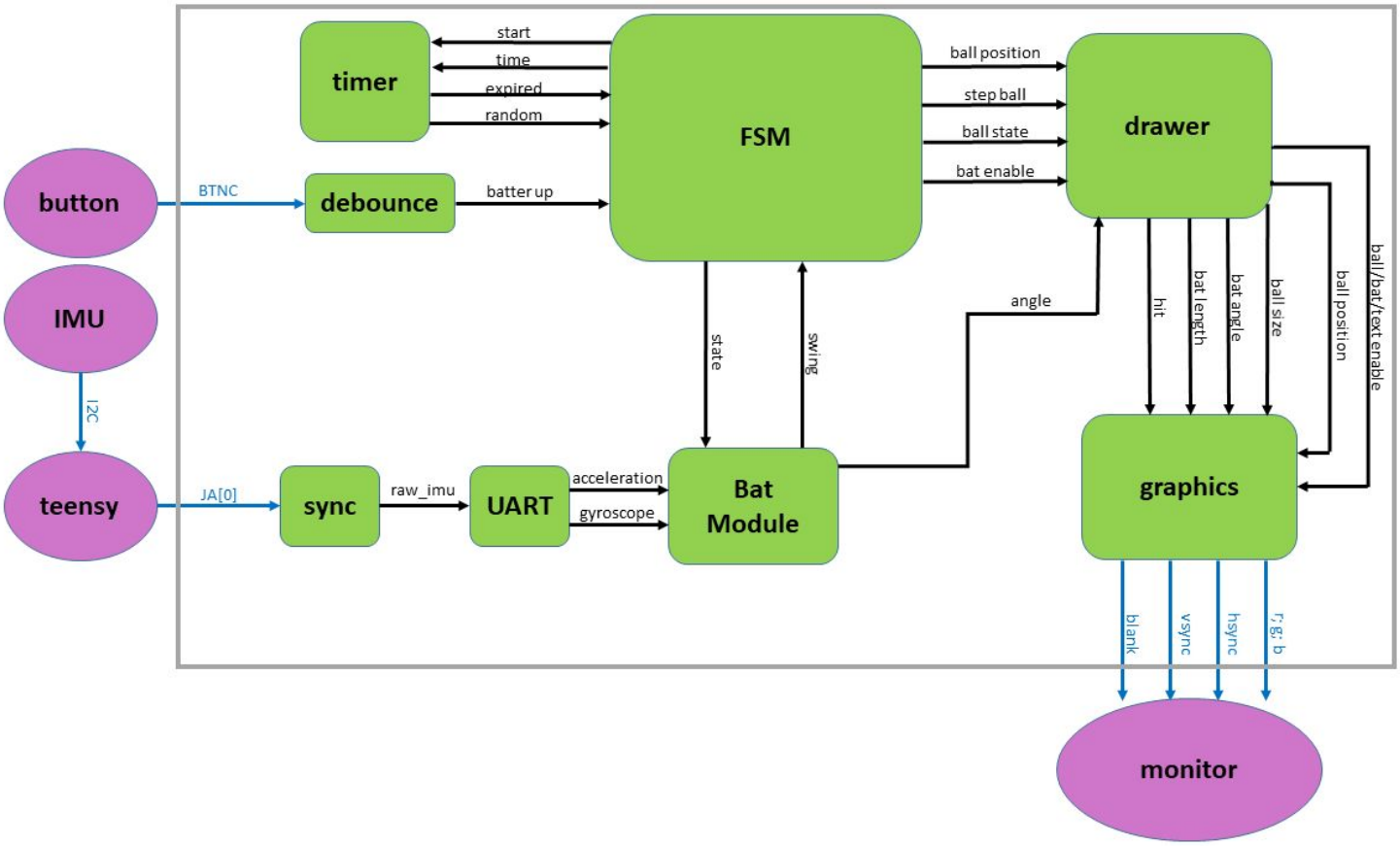
1. Wireless bat communications with the NEXYS4.
    - **P2 -** Instead of having a wired connection from the IMU, data can be sent via Bluetooth. We expect this to involve a microcontroller and additional Bluetooth receivers.

2. Improved gameplay interface.
    - **P2 -** Include softball-themed backdrop on screen.
    - **P2 -** Add sound effects for ball hits and misses.
    - **P2 -** Make game playable on TV screen.
    - **P2 -** Adjustable difficulty can be set by the user before each swing.
    - **P3 -** Include opening menu scene with available settings.
    - **P3 -** Rate hits and keep a score for the full game.

3. Indicate that the user is ready for the next pitch.
    - **P3 -** The user shows they would like to receive another pitch by raising the bat in a ready position.

4. Defensive mode of the game for a field player.
    - **P2 -** Have a ball graphically appear on the screen as though it is coming towards the fielder.
    - **P2 -** Fielder uses buttons on NEXYS4 to move virtual glove shown on screen to catch the ball.
    - **P3 -** The ball appears on the screen to the fielder in a corresponding trajectory to how the batter hit the ball.
    - **P3 -** Fielder uses a joystick to move the glove to catch the ball.
    - **P4 -** Fielder uses a real glove (probably with LEDs attached) that is tracked by a camera for its corresponding position.
    - **P4 -** We detect the closing of the physical glove by having two metal plates that connect and conduct electricity when the glove is closed. This would indicate when the fielder thinks they should catch the ball.

2.2 Block Diagram



3. Implementation

3.1 Physical Bat Interfacing

Interfacing with a physical bat will be a crucial component of making the softball game playable and exciting. To make it as realistic as possible, we will use a real softball bat with attached sensors to measure speeds and acceleration of each swing. For our initial implementation, we plan to use an MPU9255 9-axis accelerometer, secured to the end of the bat with a wired connection to the FPGA board, providing communication channels and power to the IMU. To avoid getting lost in the details of an I2C or SPI protocol, we will channel all raw data from the IMU through a Teensy 3.2, which will output all readings serially to an FPGA input pin. The `UART` module will handle reading the stream of inputs and outputting the most recent readings to the `IMU` module for calculations.

Given enough time, we hope to additionally implement wireless functionality to allow for more mobility of the player, however it may also increase the latency of the system. We would use a fast bluetooth transmitter, such as the Blue SMiRF from Sparkfun, which can go up to 115200 baud rates, to relay the information to the Teensy and include a receiver on the FPGA board to communicate with the rest of the system.

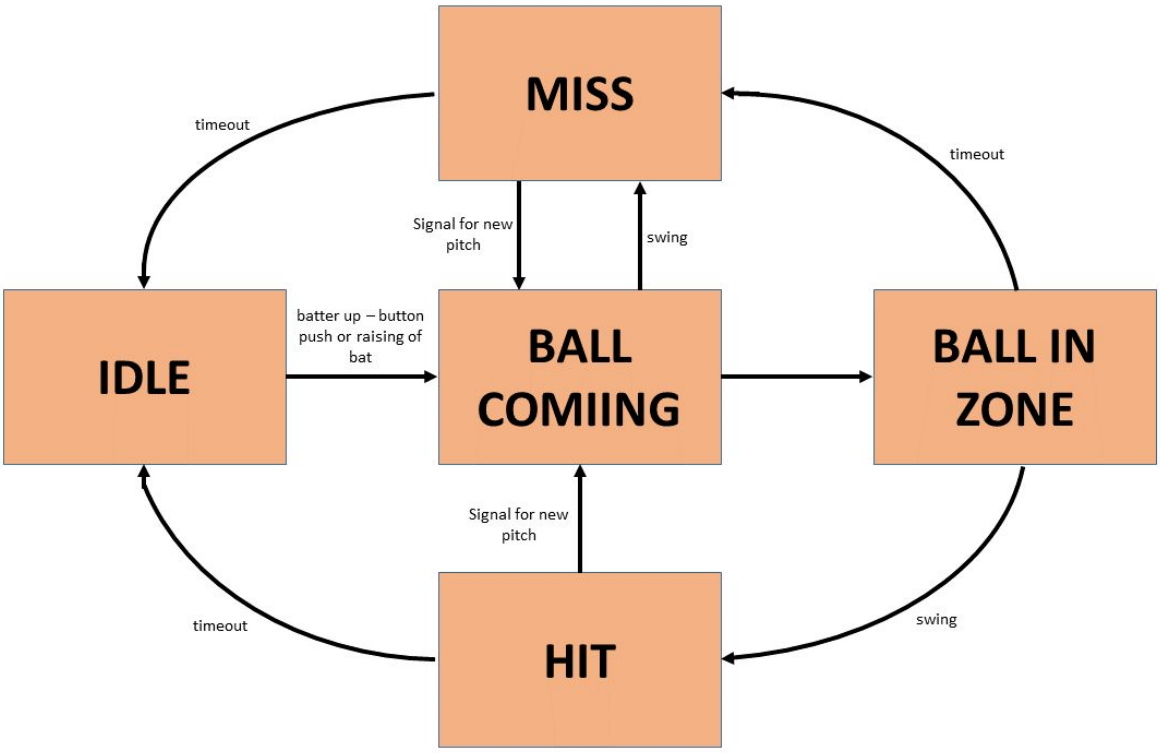
3.2 Bat Swing Calculator

The `bat_module` will serve to translate accelerometer data into meaningful information that can be processed by the main program. It receives synchronized acceleration and gyroscopic information recorded and transmitted by the IMU from the `UART` module and performs the necessary physics calculations to get key metrics of a swing. The most important readings that need to be captured are the swing timing and exact angle at the moment the bat intersects the plane of the ball. This can be calculated from the inflection point in the vertical position of the tip of the bat, which requires integrating the speed over the time of the swing and knowing the orientation of the IMU with respect to the real world. At the moment the bat reaches it's lowest point in the swing, a signal is sent to the FSM and the angle of the bat is stored, which will be used in hit calculations by the drawer module.

Although in our first approach a pitch is initiated by the press of a button on the FPGA, we would want a raised bat to indicate that the batter is ready for the ball. This feature will be implemented if we are able to get the IMU functioning properly and are able to easily read and interpret data from it.

3.3 Gameplay

The main `FSM` controls the game sequence and manages all other modules. With support of the `timer` module (modified from the car alarm lab implemented earlier in the semester), it initiates the game setup from a pseudo-randomized state by taking the least significant bits of a counter at an arbitrary point in time. We may include a difficulty setting, which determines the speed of the ball and time frame to swing, that the FSM handles by sending a step_ball signal at a certain frequency and setting the corresponding times for the timer. The main states and transitions within the FSM are depicted below. In each state, the outputs are updated to reflect all currently known values. This FSM allows for expansion of the game to include a fielder to virtually catch the ball at a location determined by the hit trajectory. Although a fun addition, this expansion will only be implemented once all other modules are functional and have met the expected goals.



3.4 Hit Calculator

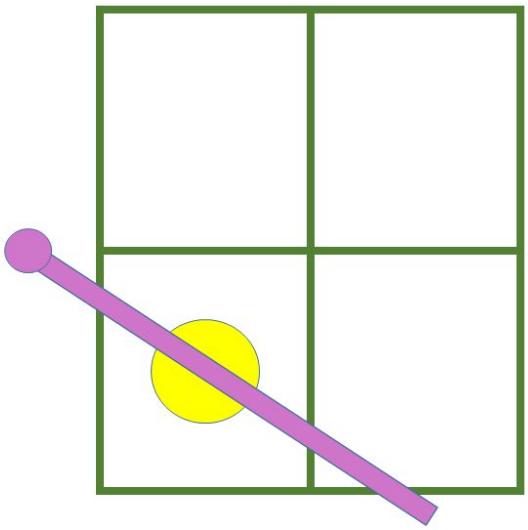
Once a swing is detected and the angle of the bat is known, the `drawer` module will have enough input data from the ball and bat states to determine how the ball was hit. It will output information on the geometry of the ball and bat to the graphics module for an accurate display, and it will return information about the hit. The metrics we consider important for a hit are vertical position (determined by angle), horizontal position (determined by timing), and speed (if we get our IMU readings accurate enough to detect such variations).

3.5 Graphics

To make the game playable and fun, we will include visual cues on a screen of when to hit the ball and feedback on the results of a hit. The `graphics` module takes inputs from the drawer on the position and size/orientation of the ball and bat. It then creates simple blobs to display on the screen. We will follow pixel representations similar to the pong game implemented earlier for a lab. Because the ball is represented as a circle, we may need to delay the output by a few clock cycles to calculate multiplications, but this lag will not be noticeable by the player because it is accounted for when measuring the timing of the swing. The display also includes a segmentation of the screen into four main quadrants to provide an indication to the player of different pitch locations requiring different swing angles. There will also be some form of feedback reflecting the success of a hit and plausible outcome of the swing either as a score counter or hit/miss indicator light as well as expected trajectory of the ball. The display screen

will primarily be a lab monitor, however, if we are able to implement a scaling factor to make the ball and bat still look realistic on different screen sizes, we would make the game playable on the TV screen as well. Additionally, we may include features such as a ballpark backdrop, sounds, or game-like menu screen if we have enough time, but our priorities are to optimize the physical bat swing detector. Given that our graphics are minimalistic and we are not storing much data about previous swings, memory will not be a main consideration for this project.

Possible Strike Zone Example Graphic:



4. Testing

4.1 List of Modules

- `UART`
- `Bat_module`
- `FSM`
- `Drawer`
- `Graphics`
- `Timer`
- `Debounce`
- `Synchronizer`

4.2 External Components

Essential:

- MPU9255 ($5)
- Teensy V3.2 ($20)
- Softball bat (borrowed from MIT Varsity Softball Team)
- Display monitor (in lab)
- Nexys4DDR board (in lab)

Additional:

- Blue SMiRF Silver- fast bluetooth module transmitter and receiver ($25 each)
- Joystick (in lab)

4.3 Design Verification

To keep our design modular and testable without having everything complete, we can separate the design into main components that can be tested individually.

4.3.1 Physical Bat Interfacing

To test UART communication between the FPGA and Teensy, we can have the Teensy output arbitrary data and verify that the FPGA is able to read back that data and separate it into meaningful chunks. Ideally, we would use sample data read from the IMU to verify the functionality of that specific process. Using this same technique, we will be able to test how well communicating over Bluetooth works and whether the data is still accurate enough for our purposes.

4.3.2 Bat Swing Calculator

For the bat swing, it is necessary to have valid IMU data read into the FPGA before being able to verify that our calculations based on that data are correct. To avoid relying on the success of the UART communication and without needing to implement an I2C protocol, we will first demonstrate the functionality of all calculations in a higher-level C-like language written directly on the Teensy. Once that is shown to be functional, we can transfer the logic to the FPGA. Once the communication is functional, we can test the full bat module. Alternatively, if communication is not yet ready, we can try to extract and save readings from a real bat swing and use those as data inputs for a simulation of the IMU calculator module.

4.3.3 Gameplay

Gameplay aims to test correct transitioning in the sequence of play, targeting the FSM module. Because we will be building off of an existing and working timer, we can use that to help signal

the correct times to transition, while providing necessary inputs via simulation or by button presses. The output will be matched to expected state transitions given the inputs.

4.3.4 Hit Calculator

Testing the whole display will require proper outputs from the drawer and a correctly implemented graphics module. We can start by verifying the outputs of the drawer module being correct, although they may be hard to judge without any visual aids. By inputting general ball/bat state and easy-to-track update values (time steps, bat angles, etc), the drawer module must calculate the correct size/orientation of each object. We will verify these outputs both numerically, to make sure they are within reasonable ranges, and visually, to make sure they look correct on screen once the graphics system is fully functional.

4.3.5 Graphics

The graphics can be tested with a series of plausible gameplay situations, described as simulation inputs. The module requires only ball position and size, bat angle and length, and enable signals for all objects. Providing reasonable inputs allows us to step through a mock game sequence without a fully implemented FSM.

5. Timeline

This is meant to give a rough timeline of what modules should be worked on each week. As we begin working, we will make harder deadlines for ourselves each week, but this is meant to give a rough idea of the rest of the semester.

- Week of 10/30
    - IMU interfacing with teensy and NEXYS4
        - Implement data flow from physically swung bat to modules.
    - Ball and bat graphics
        - Implement with fake bat and ball positions and data.

- Week of 11/6
    - Bat physics
        - Figure out how to determine the timing of when the swing would hit the ball in a real game.
        - Determine bat angle through the strike zone with data from the gyroscope.
    - Timing
        - Implement the universal timing of all modules including the bat so that the resulting swing timing data is accurate.

- Week of 11/13
    - Hit Calculating
        - Graphically implement real data from the bat and show how the ball would be hit.
        - Determine the trajectory of a hit ball.
    - Gameplay
        - Implement the FSM so that all modules are connected with correct timing.
- Week of 11/20
    - This is the week of Thanksgiving, so we realistically expect to be in lab much less time.
    - Test the existing parts and determine which stretch goals should be worked-on.
- Week of 11/27
    - Work on stretch goals
    - Wireless Connection
        - Instead of having a physical wire connected to the bat, try to transfer data wirelessly.
    - Glove/Fielding Modules
        - If time permits, implement the fielding modules to play the defensive parts of virtual softball.
- Week of 12/4
    - Finishing Touches
        - Incremental enhancements
        - Small changes made to improve the user experience
- Week of 12/11
    - Checkoff by 12/11
    - Video filming on 12/12
    - Finalize the report due on 12/13

6. Responsibilities

Subsystem	**Designer**
Physical Bat Interfacing	Melinda
Bat Swing Calculator	Both
Gameplay	Melinda
Hit Calculator	Katie
Graphics	Katie