Dec1701

Client:
Accu-Pop L.L.C.

Faculty Advisor:
Dr. Ahmed Kamal

Team Members:
Andrew Zellar - Team Leader
Jordan Young - Communications
Stefan Kraus - Webmaster
Josh Wassenaar - Key Ideas Holder

Contact:
dec1701@iastate.edu
http://dec1701.sd.ece.iastate.edu

2/17/2017 - Version 1

Accu-Pop

PROJECT PLAN

Contents

1 Introduction

Accu-pop is a product that we are designing that will be used to measure a catcher's pop time, so that scouts can accurately assess catchers.

1.1 Project statement

In baseball, "pop-time" is the time it takes from the moment that a baseball hits a catcher's mitt to the moment the ball hits the glove of the second basemen. This is one of the most important ways the skill level of a catcher can be assessed by a scout. The most commonly used method for obtaining pop-time is by using a stopwatch to time the throw it. This is a very inaccurate process and renders results with an error of .2 seconds, which is a very large error when the average pop-time is around 2 seconds. Therefore, a better method for accurately recording the pop-time is needed. The goal of this project is to accurately measure the time it takes for a catcher to throw a ball down to second base, and then display that time on an app and website.

1.2 Purpose

This product will make significantly more accurate readings of pop-times of catchers, which will help baseball recruiters to better assess and compare the skill level of catchers at clinics and showcases. A more accurate reading of pop-time is necessary because recording pop-times by hand is inaccurate and can have an error of up to .2 seconds (10% of the average pop-time). This error can be the difference between an elite catcher and a sub-par catcher. It is necessary to return accurate readings so scouts and other interested parties can correctly judge catchers based on their pop-times. Given that this is phase two of this project our main purpose is to further increase the accuracy of our system to better record catcher's pop-times.

1.3 Visual Description of a Pop-Time

In the image depicted below, Accu-Pop is aimed to determine the time of the 'Second throw.'

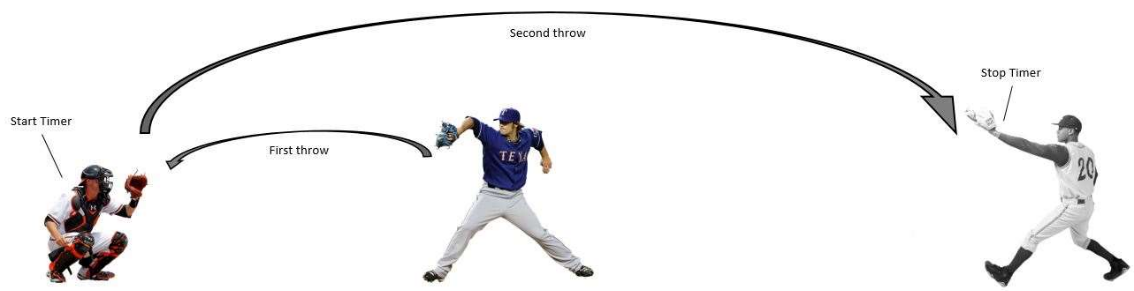

Figure 1: Visual Depiction of a Pop-Time [1]

1.4 Goals

This project has the following required goals:

1. Design and implement a complete prototype, accurate to within 1/100th of a second.
2. Develop a phone application program (available on iOS and Android platforms) and a

live-updating website.
3. Clear documentation of all products and code.

Stretch Goals:

1. Measurement device
    a. The device will be suitable for use in-game
    b. The device can also be used to measure the time from pitcher to catcher
    c. the cost of the device components should not exceed $70
    d. The device has a way of receiving software updates automatically when connected to a computer
2. Web, Android, and iOS application
    a. Users should be able to track pop-times over time in a graph view
    b. Users should be able to share pop times for players with other users
3. Stretch Goal: measure time from pitcher to catcher.

2 Deliverables

The following are the expected deliverables by the end of the project:

1. A completely functional prototype, accurate to within 1/100th of a second.
2. A phone application program (available on iOS and Android platforms) and a live-updating website.
3. Stretch Deliverable: measure time from pitcher to catcher.

Requirements:

1. Technical Requirements
    a. The device should measure the pop time with a reliability of 90%
    b. The device should measure the pop time with a margin of error not exceeding 5 millisecond
    c. Users should be able to view the data transmitted from the measurement devices across web and mobile platforms.
    d. Users should have a way to save recorded pop times to the server and retrieve the times from other devices
    e. The device should have a way of transmitting data via bluetooth to a central control
2. Non-technical Requirements
    a. The device should not prevent players from moving freely
    b. The cost of the components of the device should not exceed $100

3 Design

This project is phase 2 of an already existing project. In phase 1 the project was successful in making pop-time readings to roughly a tenth of a second, additionally they have basic functioning applications for both Android and iOS.

For the continuation of this project, we have the option to improve on the previous group's design or use a completely alternate design of our choosing. We have come up with a few options, the benefits and drawbacks of each options are considered below:

1. **Improving on the Previous Design**
    a. Benefits
        i. The project structure is known, well documented, and shown to work in at least a minimal capacity.
        ii. Hardware components are already bought and installed with the recording software
        iii. Most of the work would be a refinement of the current technology
    b. Drawbacks
        i. It may not be possible to improve the previous design to meet the client's standards
        ii. We would have to upgrade the measurement devices to use Bluetooth 5
2. **Camera Sensor and OpenCV**
    a. Benefits
        i. Plays to the software strengths of the group
        ii. Development would be primarily software-based, which means less hardware testing and fewer hardware requirements
        iii. Passive collection of data - does not require players to wear special equipment
        iv. Allows deferral of cost of camera to the customer - if allowed, this would mean that cost of the product to produce would be ONLY server upkeep
    b. Drawbacks
        i. The camera would have to meet certain requirements in order to satisfy the client's requirements (discussed below).
        ii. If we must include the cost of the camera, we will most likely go over budget
3. **Sound Sensor(s)**
    a. Benefits
        i. Fairly cheap sensor
        ii. Strikes a good balance between software and hardware development load
        iii. Passive or almost-passive data collection, depending on where the sensor must be located in relation to the player
        iv. Sound processing is very fast, meaning the results will require relatively little processing and will therefore return quickly with a small processor load
    b. Drawbacks
        i. Would not work at all for measuring the time from pitcher to catcher
        ii. For one solution, device synchronization is required which the previous team struggled with

iii. Errant sounds would interfere with the results, so we would either have to find a way to control for those or make a requirement that the user does not create sounds that would be picked up by the sensors
iv. Sounds has a relatively slow propagation through air (340 m/sec) and it the devices are far enough apart there will be significant time measurement errors

4. **Pressure Sensor**
   a. Benefits
      i. Cheap sensor
      ii. A good balance between hardware and software
      iii. Likely one of the easiest signals to process, making results most likely to be accurate and return fast
   b. Drawbacks
      i. Somewhat invasive, as it would require the player to wear sensors in their gloves
      ii. requires multiple sensors or very specific catching area in order to trigger
      iii. Durability may be an issue, so the sensors would have to both have good protection and be easily replaceable
      iv. sensitivity of pressure sensors may cause false readings

3.1 Previous work/literature

This project is a continuation from a previous senior design group. The plans, diagrams, hardware, and code will be used as a starting point for this project.

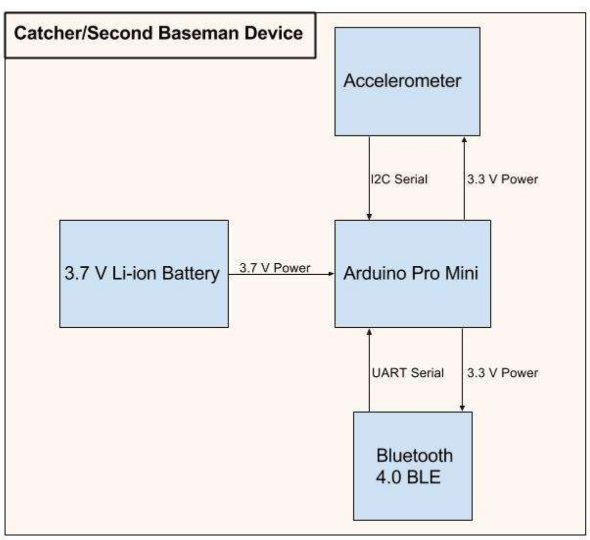

Figure 2 Final Block Diagram from previous group [1]

3.2 Proposed System Block diagram

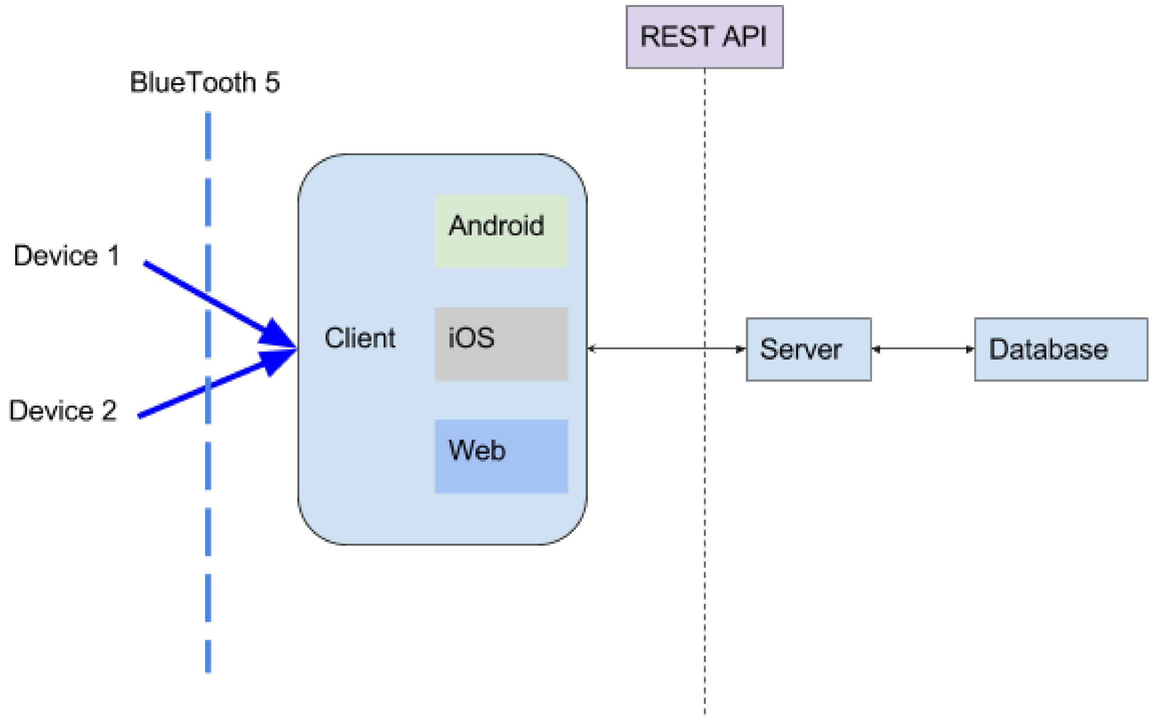

Figure 3 Top level full system diagram displaying how the devices will communicate with the applications to display the information on multiple platforms in real time.

3.3 Assessment of Proposed methods

Details of Alternative designs:

Video Camera

In order to reach the client specification of being accurate to 10 ms, we must have a camera that takes video at either 100+ fps or have a reliable way to interpolate a video feed from a 60 fps camera. A 30 fps camera will not be able to be interpolated accurately because the shutter speed will probably be too slow to get a clear image of the ball, and that would introduce too much error and uncertainty into the readings.

For a 100+ fps camera, the time that a player catches a ball can be said to be the first frame that the ball is obscured by the glove. This is simple enough and requires no additional rectification to achieve the desired precision.

At 60fps, having a maximum expected pitch velocity of 90 mph means that the ball may travel 2.2 feet per frame of video. We could interpolate the speed and position of the ball if we controlled the position of the camera and distance to the catcher, or just use a simple ratio. This would mean that the camera would need to capture the area at least 5 feet in front of the catcher.

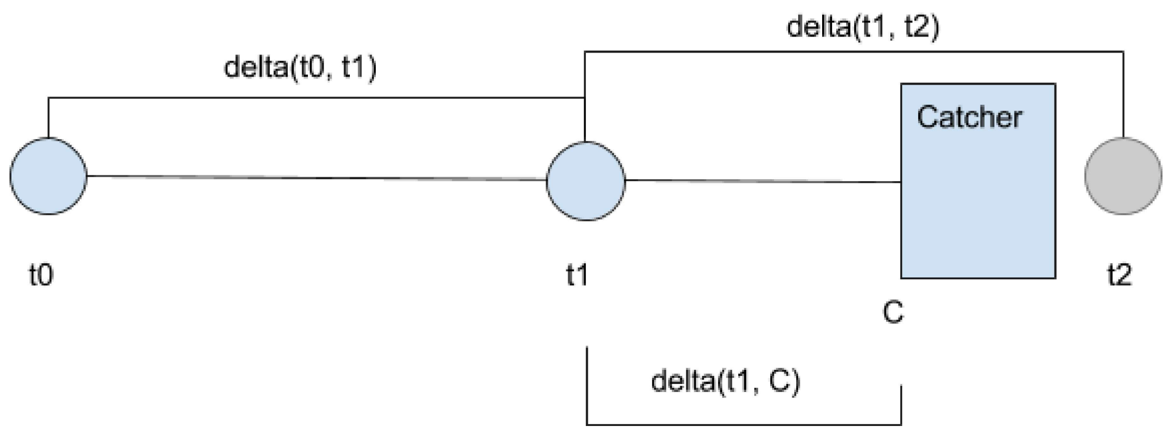

Figure 4 visualization of how measurements could be taken with camera

Visually, t2 is where the ball would have been had it not been caught, t1 is the last position of the ball before being caught that was captured by the camera, and t0 is the position before that. Then, the actual time, or split frame, at which the ball was caught can be calculated as follows:
t1 + (delta(t1, C) / delta(t0, t1))

This solution should result in very accurate readings assuming that:
- the video feeds can be synched
- the shutter speed allows for the ball to be identified consistently
- the ball is thrown consistently so that the positional requirements are fulfilled
- the catcher does not move out of position
- the client positions the camera correctly

Cost of One Raspberry Pi w/ 720p/60fps camera module: $65
Expected cost: $140-150 per unit

Alternatively, we could put the burden of supplying a webcam onto the client which would essentially negate this cost but would require a slightly different structure to the application.

This solution may also not be patentable because of its generality and use of open source software.

Sound Sensors

Solution 1: Single mic

This solution puts a single microphone (maybe omnidirectional, maybe bidirectional…) between the catcher (in the baseball position sense) and the second baseman. The mic would pick up the sound signatures of the catcher and subsequently the second baseman catching the ball.

As long as there is an equal distance between each catch and the mic, no additional rectification of the time signatures is needed.

This would most likely mean that the single mic would be higher quality. and could be plugged into a user device directly.

This also means that the solution is a pretty general one, and that could mean that there would be no way to have a patent on it.

Another point to consider is the potential error caused by the slow propagation of sound through air, thus varying distances between sensors and the glove impact can have a serious impact on the accuracy of the system.

Solution 2: two mics

Here, we would put a sound sensor attached to an arduino on each player. This would probably increase the individual accuracy of the sensors, but would require the devices to be synched. The arduino could either:

1. Send all audio data to the client app for processing

2. Only send a "start/stop" signal once a proper signature was received

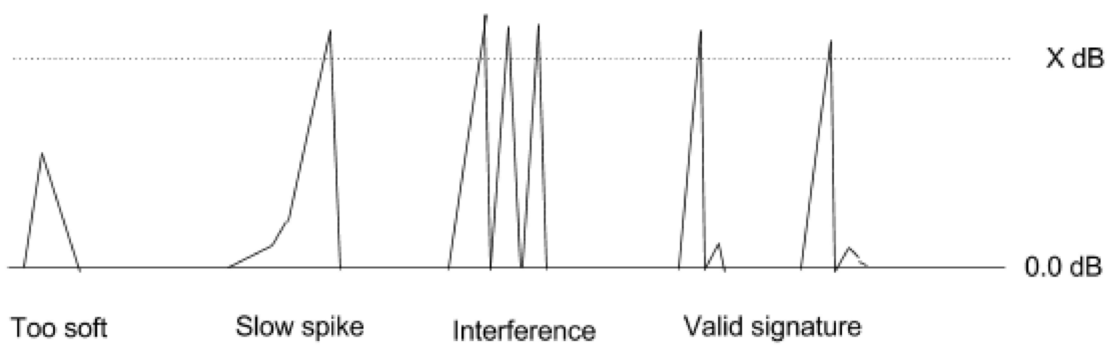

Figure 5 Examples of invalid and valid sound signatures in waveform against a TBD decibel threshold.

Problem: The speed of sound at normal atmospheric parameters is approximately 1130 feet per second. That means that it travel 1.13 feet per millisecond, meaning that approximately for every 13 inches (35 cm) the sound sensor is away from the sound source, the reading will be 1 ms off. Now, this may not be as big of a problem as long as the sensor(s) is/are equidistant from where each player catches the ball. This is a potentially difficult requirement to ensure consistency with - product design must account for this.

Expected cost of 2 arduinos, 2 microphone sensors: $70-80

Expected cost of a bidirectional or omnidirectional mic: $50-60

Pressure Sensors

The player would wear a wristband connected to one or more pressure sensors in their glove, which would trigger when a ball hit it with some force. The arduino would trigger an event at that time and send a signal to the main application. The devices would need to be synchronized.

For measuring the time from pitcher to catcher, the pitcher could put their foot on a pressure sensor and it would trigger once lifted, so this solution may fulfill that stretch goal.

Expected cost of 2 arduinos: $50
Expected cost per pressure sensor: $7

Additional costs (Ruggedization, etc.): $15-20 per system

Total: $65-70

3.4 Validation

The process for validation of this project will require us to first record the pop-time of a test throw with our product while videotaping the throw. Then we will analyze the video we take and step through it frame by frame to get the actual time of the throw. Once we have obtained our measured time and the actual time we will compare and draw conclusions based on the accuracy (or lack thereof) of our product.

4 Project Requirements/Specifications

4.1 functional

4.1.1 Hardware functional

1. The sensor needs to be reliable enough to get an accurate reading 95% of the time
2. The hardware needs to account for latency and other factors caused by the hardware to keep the error under one hundredth of a second
3. The system must connect to a wireless network and report data to a mobile device or laptop

4.1.2 Software functional

1. There must be an app for iOS and Android devices
2. There must be a website that displays the data after it is collected
3. Both the app and the website should allow for data filtering, sorting, and searching.
4. The system should allow the user to stop a pop time

4.2 Non-functional

1. The system must not interfere with normal play

2. System cost should be kept under $100 per system, preferably under $70

4.3 Standards

Coding Standards:

General:

Any method above two lines should have a docstring, and those that are below two lines should probably not exist. The specific coding standards for any language used without a single prescribed or generally followed code style guide should be kept consistent within the project. Tests are encouraged, but not strictly required in favor of quick prototyping functionality. The attitude towards code quality is as follows: "Make it work, then make it pretty, then make it scale."

Python:

Python code should be, well, Pythonic. Methods and classes should be small and simple, and deliver specific functionality within the context of their module. Classes should not have gratuitous properties on them, and dependencies should be kept to a minimum.

As far as style guides, PEP-8 should be followed: https://www.python.org/dev/peps/pep-0008/.

JavaScript (JSX / React Native):

JavaScript should be as functional as possible when possible. We will be using React Native, so the exact semantics will not be identical, but in general the app should follow JSX patterns of neatness and organization.

Since the majority of the JS will be in React (JSX) code, the airbnb JSX style guide should be followed: https://github.com/airbnb/javascript/tree/master/react.

5 Challenges

Delivering Project Specifications

The main challenge of this project will be succeeding where the previous group failed, namely, to satisfy the accuracy requirements of 10ms. We have a set of very ambitious project specifications that require a top quality product to fulfill. This means that we will have to learn from past mistakes and put forth a great deal of effort to overcome the physical and technological challenges the project poses and deliver the final product within the timeline we set out for ourselves.

Teamwork and Group Cohesion

Working as part of a team can be challenging, and being able to do so effectively takes a lot of work. We must all put forth an effort to hold one another accountable while still maintaining a positive group experience, or else we may fall apart and cease to be able to work with one another.

6 Timeline

Gantt Chart First Semester

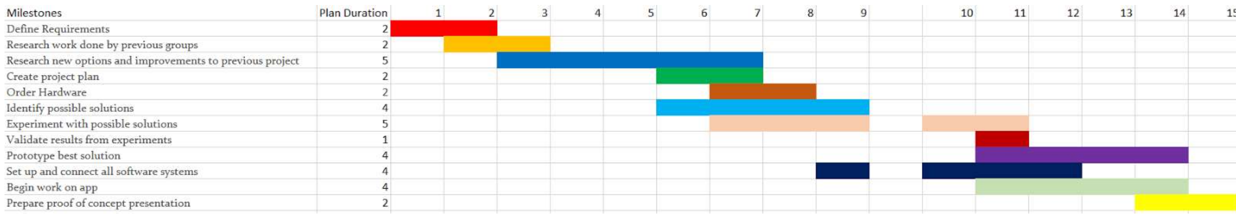

Figure 6 First semester gantt chart

Gantt Chart Second Semester

Figure 7 Second semester gantt chart

6.1 First Semester

1. Define Requirements
2. Study work done by previous groups
3. Research new options and improvements to previous project
4. Create project plan
5. Order Hardware
6. Identify possible solutions
7. Experiment with possible solutions
8. Validate results from experiments
9. Prototype best solution
10. Set up and connect all software systems
11. Begin work on app
12. Prepare proof of concept presentation

6.2 Second Semester

1. Reconfigure as a team after summer break
2. Reevaluate project plan
3. Test and refine prototype both hardware and complete system
4. Complete software implementation
5. Documentation

6. Prepare for final presentation

7 Conclusions

Accu-pop is designed to be a timing system for catcher's pop-times. Our goals are to produce an accurate, consistent, and cheap product that will accurately record a catcher's pop-time. Additionally we seek to develop fully functioning Android and iOS apps, as well as a live-updating webpage for easy access to pop-time results.

We plan to build off of the existing product from last year's group. In order to do this we must first fully understand the current status of the product as they left it. Additionally we will be researching alternative solutions, and testing all hardware that we think could potentially aid us in creating Accu-pop. We will divide the work into hardware and software components as each of these has very different functional and nonfunctional requirements. In order to complete this project in a timely manner we will follow our previously stated timeline and project plan to the best of our abilities. By following the plan we have set in place we believe in our ability to deliver a complete product by the end of the Fall 2017 semester.

8 References

Reference Number	Reference Style	Reference Title
[1]	Document	Spring-Fall 2016 Dec1601 Final Project Plan