ELECOMP CAPSTONE SUMMIT
FRIDAY, MAY 10, 2019

THE UNIVERSITY OF RHODE ISLAND

THINK BIG WE DO™

Pison iOS Integration

PISON

Team Members: Keara Cole (CPE), Alisha Mitchell (CPE)

Technical Directors: David Cipoletta, Sam Karnes, Mike Kowalczyk, Sid Srikumar | Consulting Technical Director: Brenden Smerbeck ('17)

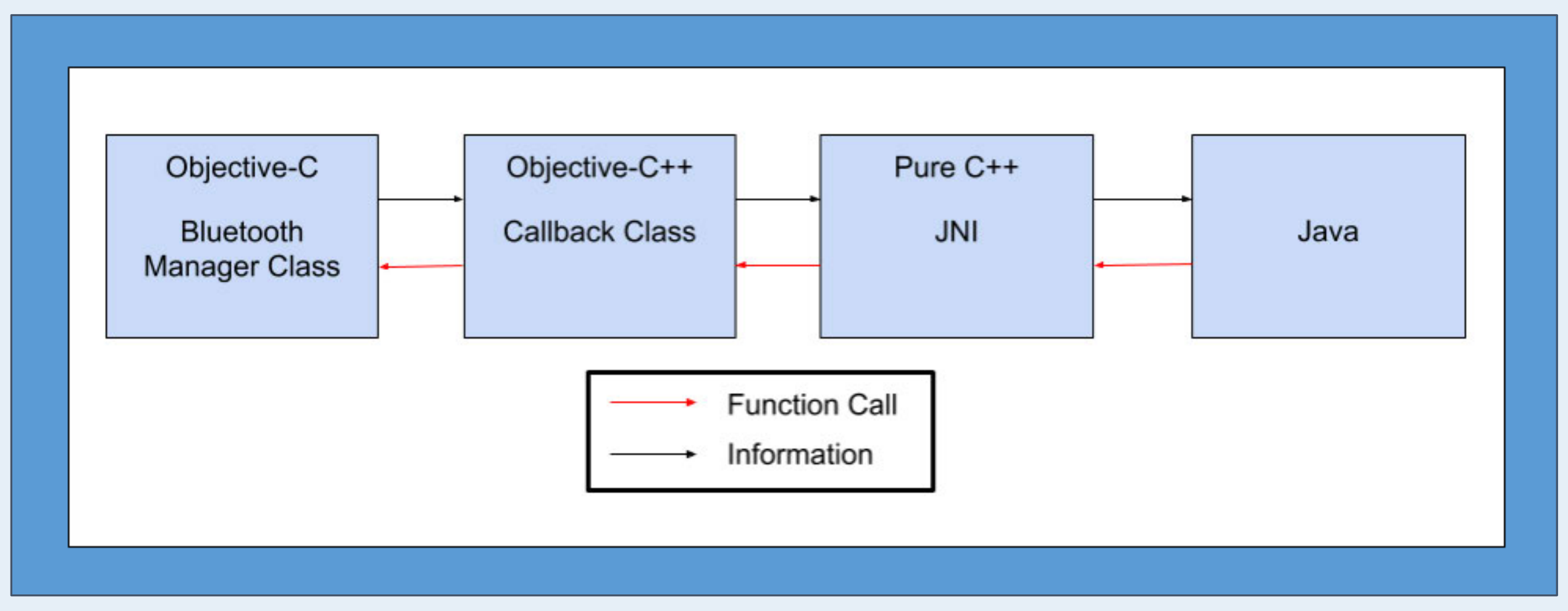
Project Motivation

The objective for the Pison iOS integration project is to extend Pison's existing Windows and Android systems to encompass macOS and iOS systems. Device compatibility will increase Pison's competitive market-space; by incorporating the largest device platforms, Pison can offer their device to a wider range of customers. Ultimately, this equates to greater profits and overall recognition within the wearable and assistive technology markets.
Another motivation of the Pison iOS integration project is saving Pison's engineers time by writing code that is portable and compatible with Pison's pre-existing Java system. Without interfacing with Pison's existing system, every addition would need to be rewritten for the macOS and iOS platforms. Instead, by adding Bluetooth functionality and bridging it with their existing system, we allow Pison to transfer all of the work they've done to the macOS and iOS platform. Saving time while producing cross-platform software provides invaluable resources for the development of Pison products.

Key Accomplishments

Overview of Key Accomplishments

Implemented a Bluetooth manager in Objective-C that utilizes Apple's Core Bluetooth Framework to gain access to Apple's Bluetooth hardware. Then, that entire code is wrapped in Objective-C++, so that it can be called from by a code written in pure C++. Finally, the team implemented the Java Native Interface file in pure C++, which allows Java to call the methods inside of the Bluetooth manager code. A visual representation of the function call process is shown below.



Specific Details of Key Accomplishments

- **Objective-C:** Learned Objective-C to implement Bluetooth manager code. Created programs to check the team's understanding of the features unique to Objective-C.
- **Low Energy Bluetooth (BLE):** Researched Bluetooth LE and how it communicated with devices. Implemented code to communicate with a Low Energy Bluetooth device.
- **Java Native Interface (JNI):** Learned how to interface Java and Objective-C. Implemented a program to demonstrate how to use JNI to call to Java. Wrote a native Java program, compiled it to create a native library. Wrote C++ code that defines the Java methods. See **Fig 1** for more information about the JNI process.
- **Noble:** Noble is an existing platform that utilizes many aspects of the Core Bluetooth Framework that the project relies on. Using Noble and Node.js, a code that scans and connects to local peripheral devices was created. The scanning code was the starting point for using core Bluetooth in this project.
- **Setup Bluetooth Manager (Objective-C):** The initial function allocates the space in memory for the Central Manager and then initializes it. It sets all helper variables and booleans to their appropriate values, which later ensures the code is happening sequentially, the central manager is not busy, the information is retained through strong pointers, etc.
- **Scan and Connect to Device (Objective-C):** To circumvent the need for wait functions and additional booleans, the scan function repeatedly calls itself until the manager has hand enough time to power on. The connect function is then invoked, and capable of getting a list of known peripherals by their identifiers, and will automatically connect to Pison devices based on their UUID.
- **Populate Services and Characteristics (Objective-C):** Apple provides two functions that are automatically invoked upon a service or characteristic of the device being discovered. Because the Java code needs something to call when it's ready to get the services and characteristics, however, invokable helper functions had to be added to pass this information to the JNI.
- **Read, Write, and Notify (Objective-C):** Reads the Uart transmitter characteristic value and prints it to the console so that the user is able to see what is being received. Takes a value from the user and writes it to the Uart receiver characteristic. Monitors the Uart transmitter data and detect changes in data and notifies the user of changes. See **Fig 2** for more information about the Bluetooth manager dependencies.
- **Callback class for Objective-C++ library:** An Objective-C++ code that wraps the Objective-C code so it can be called from the Java Native Interface implementation which is written in pure C++. Implements functions that call all of the Bluetooth manager class functions. Implements functions to read the Objective-C objects because the pure C++ JNI code cannot have any link to the Objective-C code.
- **Setup Bluetooth Manager (C++/JNI):** Initializes the central manager from the Bluetooth code and the Java Virtual Machine pointer. Calls the setup function from the Callback class and reads the state of the central manager. Change the Bluetooth status to Bluetooth manager state. To see how the files are organized to make a JNI call possible, see **Fig 3**.
- **Scan and Connect to Device (C++/JNI):** Makes four calls into the callback library to scan, check if devices have been discovered, and read their UUID and device name. Also invokes two helper functions that convert the object to Java.
- **Populate Services (C++/JNI):** Converts the native device to a Java device, and calls into the callback library to populate services, which in turn invokes the method that waits within the Objective-C code for a service to be discovers and returns it.

Anticipated Best Outcome

The anticipated best outcome is to have a fully functioning macOS platform. The macOS platform will be a dynamic library that will contain method definitions for Pison's existing Java System. The Pison engineers will load the dynamic library into the Java system and be able to communicate with the macOS Bluetooth hardware. The library will contain a Bluetooth manager class written in Objective-C, a callback class written in Objective-C++, and a Java Native Interface written in pure C++. The dynamic library will give the user all the same functionality as Pison's existing Windows and Android systems.

Project Outcome

Results towards achieving the Anticipated Best Outcome will be presented at the Summit.

Figures

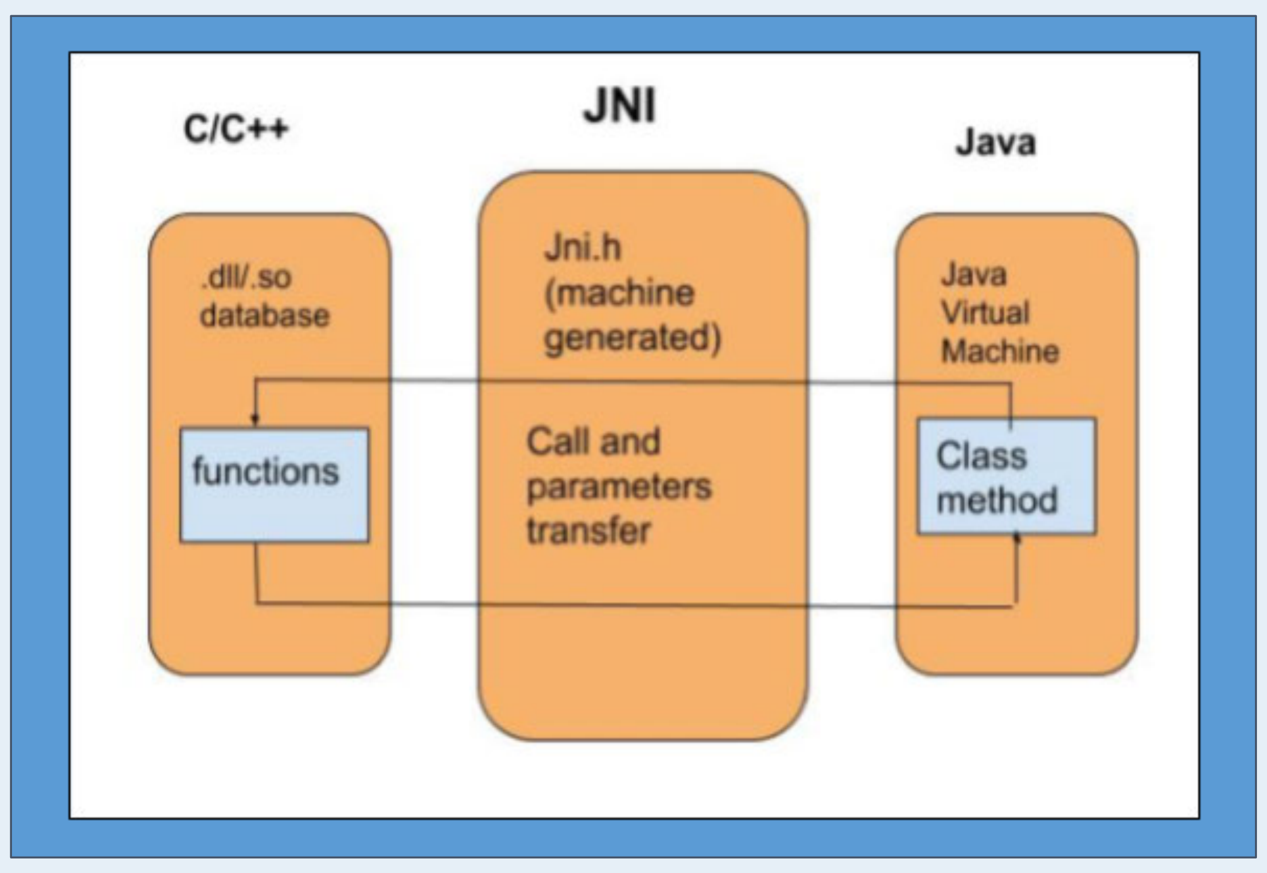


Fig. 1: Java Native Interface (JNI) process. The diagram shows how the JNI passes information as well as calling methods from Java and functions from C++.

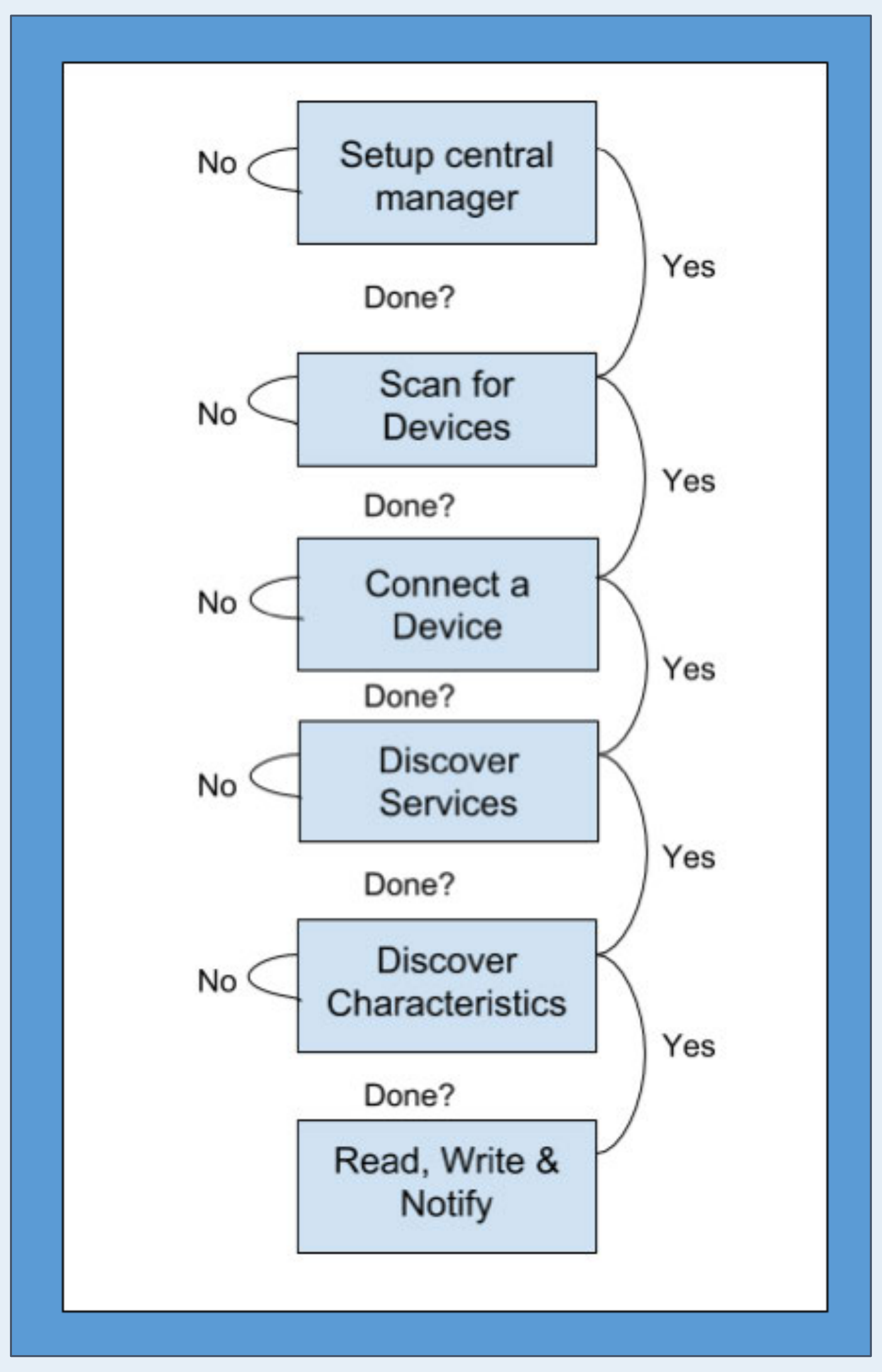


Fig. 2: Flowchart of the Bluetooth Manager Class. This shows the dependencies of the central manager which controls the Bluetooth hardware.

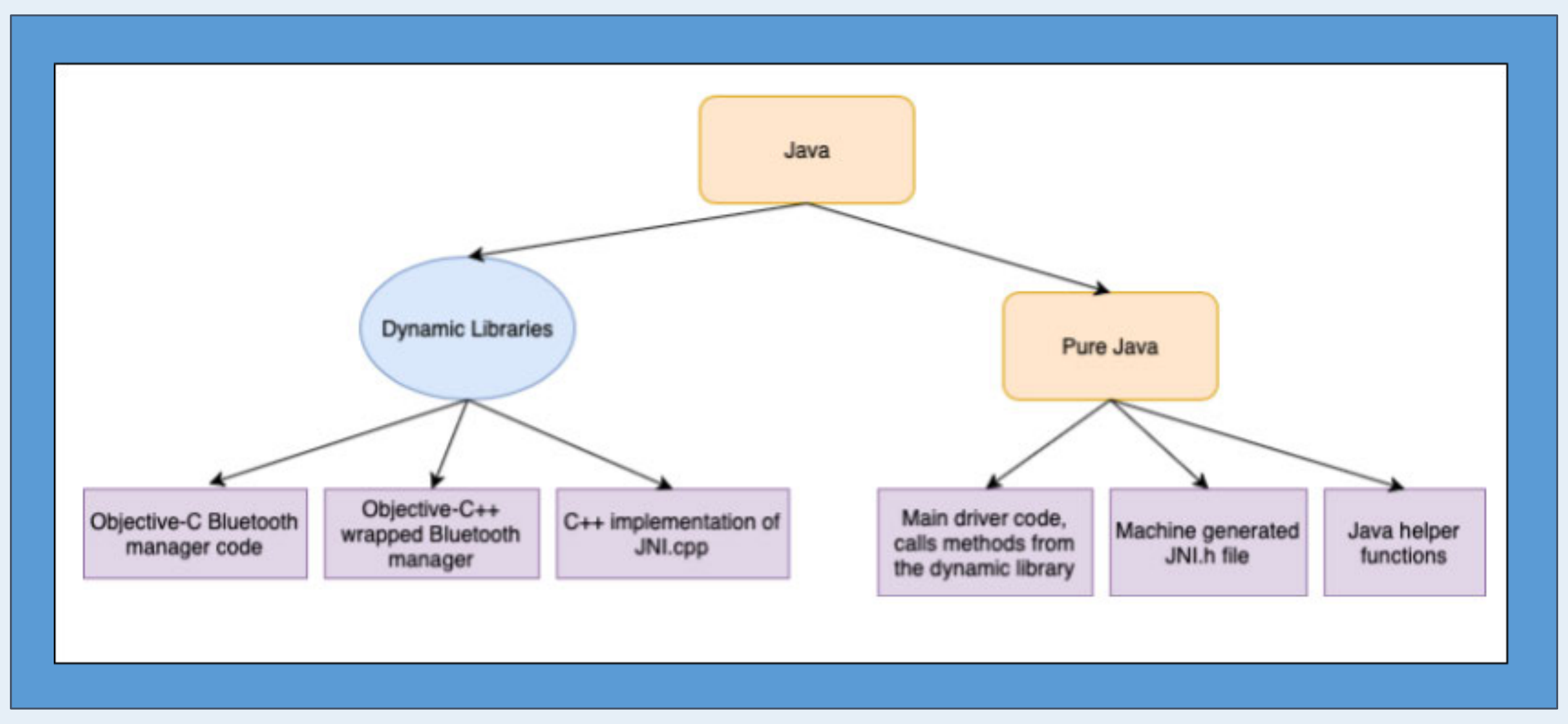

Java
Dynamic Libraries
Pure Java
Objective-C Bluetooth manager code
Objective-C++ wrapped Bluetooth manager
C++ implementation of JNI.cpp
Main driver code, calls methods from the dynamic library
Machine generated JNI.h file
Java helper functions

Fig 3: File hierarchy within the Pison iOS Integration Project. Displays the division of major files and the dynamic BLE library.

ELECOMP Capstone Program Director: Professor Harish R D Sunak

email: sunak@ele.uri.edu telephone: 401-874-5859 ELECOMP Website: https://uri.edu/elecomp-capstone