SensorPack Android Application Documentation

Joseph Maestri

July 2016

1 Introduction

- **Function of sensor package:** use sensors to collect data from environment on temperature, conductivity, depth, light, and heading. A gyroscope and accelerometer are also included in the package.
- This Android application allows the user to communicate with the Arduino over Bluetooth and download data in real time or from files on an SD card. This data is then displayed on the Android screen in a human readable format.
- The display of this application includes graphs for temperature, conductivity, depth, and light. It also includes a rotating compass for the heading data display, and a rotating seaperch for the gyroscope data display (displacement was calculated from this data, then displayed, the raw data is not shown).

2 Installation instructions

- To install the Android application on your device, first go to:
`https://github.com/Xzib1t/SeaSense.git`
and click "Clone or Download", then "Download ZIP"
- Once the ZIP folder is downloaded, extract the files to a desired location
- Next, download Android Studio from:
`https://developer.android.com/studio/index.html`
and install it on your computer
- Open Android Studio and click on "Open an existing Android Studio project"
- Navigate to the location you extracted the ZIP folder to. Inside of the Android folder you will find the main folder/project for the SensorPack application. Open this project and wait for it to build

2.1 Enabling developer mode

- Skip to the next subsection if you already have developer mode enabled
- Turn on your Android device, go to Settings ⇒ About ⇒ Software information and tap build number seven times

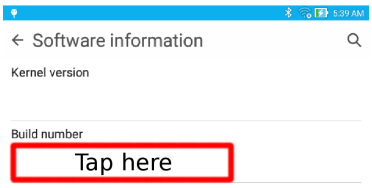
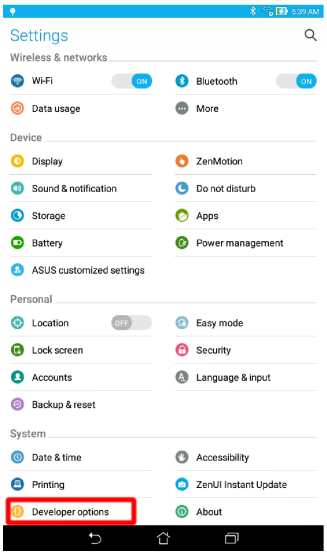


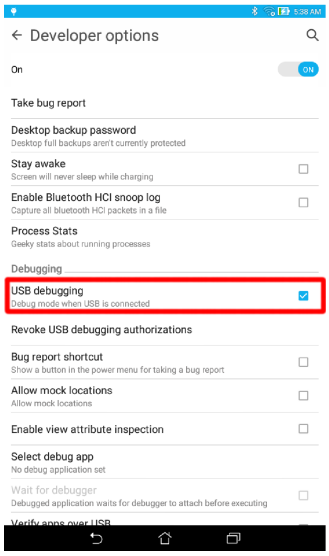
Figure 1: Enabling developer mode

2.2 Finishing Android setup

- Open developer options, then turn on USB debugging



(a) Finding developer options



(b) Enabling USB debugging

Figure 2: Finishing Android setup

- Connect your Android device to your computer with a USB cable
- Click the green arrow and then your device to build and install the application

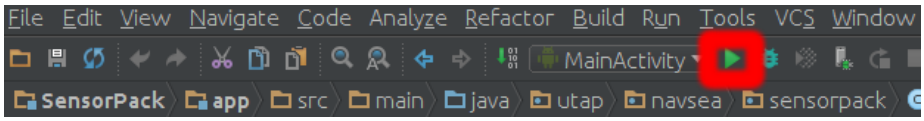


Figure 3: Uploading program to Android device

- When it is done installing, disconnect your device and **turn off USB debugging again for security**

3 Use instructions

- Open SensorPack application
- Follow the instructions that appear on start, or follow the below instructions with figures

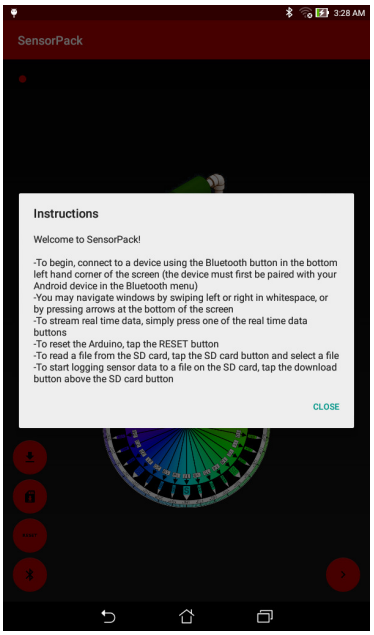


Figure 4: Application instructions

3.1 Connecting to Arduino

- Tap the Bluetooth button in the bottom left hand corner of the screen, then select a Bluetooth device from the menu that pops up. Continue attempting to connect until a “Connected” message appears.

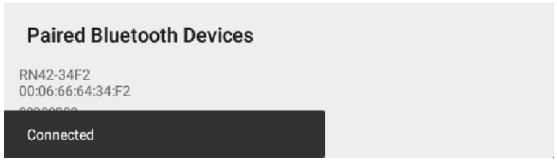
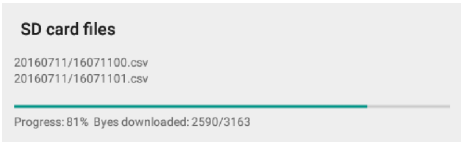


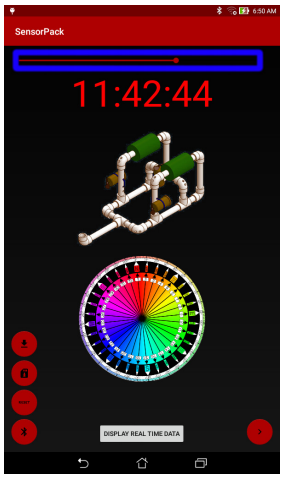
Figure 5: Android connected to device over Bluetooth

3.2 Additional commands on the main screen

- Tapping the "RESET" button at any time will send a command to the Arduino that resets it
- Tapping the SD card button will bring up a clickable list of files on the SD card. Clicking any of these files will download them to the Android device.
  - To access downloaded data, simply swipe or use the arrows to view the graphs on the other screens, or move the slider on the main screen to scroll through the compass and gyroscope data with timestamps



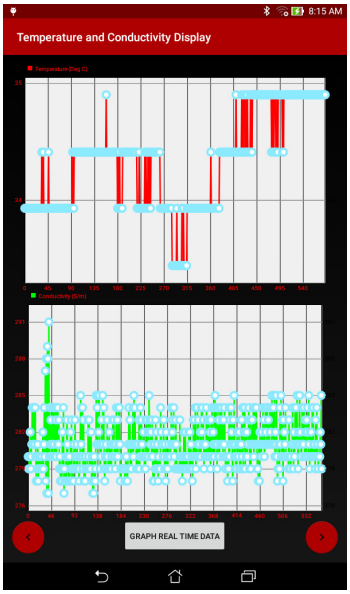
(a) Downloading a csv file



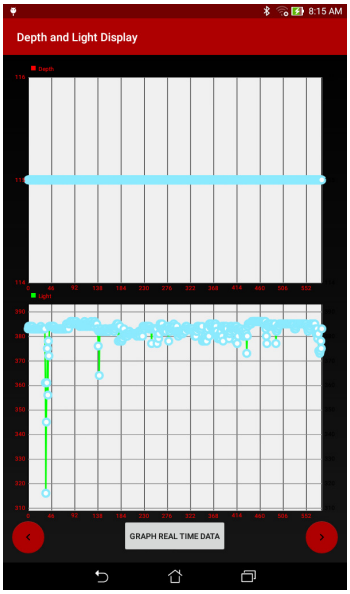
(b) Slider (highlighted in blue) used to scroll through data

Figure 6: Getting and viewing csv files from SD card

- To view graphs of data from the downloaded csv file, simply switch the view to either of the graphing screens, and the data will be automatically graphed



(a) Temperature and conductivity plot



(b) Depth and light plot

Figure 7: Graphing csv files from SD card

- To log data to a csv file on the SD card, tap the button above the SD card button (the download button) and the Arduino will begin the logging process
  - Tap the button again whenever you want to stop the logging process
  - You can access the file by downloading it from the SD file menu in the app

4 Code overview

The following is a high level code and file purpose breakdown.

4.1 File overview

The main files that you will edit and read if you work on this application are briefly explained below. This readme only covers the main java and layout files for the application, as these are the core of the app. However, if you wish to change large portions of the app and add new activities, you will need to create your own files as well as change some existing files that are not explained in this guide (such as the AndroidManifest.xml file).

- **MainActivity.java** - This is the main "activity" for the application. It is

the code that programmatically controls what you see on the first screen when you start up the application

  - Accesses and manipulates the contents of the layout files **activity_main.xml** and **content_main.xml**
  - Displays and populates the Bluetooth device menu and the SD card contents file menu
  - Holds the buttons that:
    * Tell the Arduino to start and stop logging data to a file on the SD card
    * Get the file list on the SD card and selectively download files
    * Reset the arduino
  - Rotates the images of the seaperch and compass based on incoming real time data and csv file data
  - Changes time display based on csv file data
  - Runs connections checks, button sync checks, and incoming data checks
  - Displays instructions for the app
  - Handles downloading and connecting over Bluetooth in AsyncTasks (essentially threads that run in the background)

- **Bluetooth.java**
  - Handles connecting to a Bluetooth device and passing the socket to other classes

- **Parser.java**
  - Gets data from the SD card about file sizes, quantities, and names
  - Downloads a selected file from the SD card (selection handled in MainActivity.java) and parses the data into the different data categories
  - Parses real time data also using commas as delimiters
  - Handles timeouts anywhere a blocking InputStream.read method is used
  - Acts as central storage for the sensor data
  - Returns sensor data to other portions of the app when requested

- **Commands.java**
  - Handles sending all of the available commands, stored and sent as ASCII code, to the Arduino

- **Graph.java**

  - Handles setting up graphs and plotting for **TempCondActivity.java** and **DepthLightActivity.java**

- **TempCondActivity.java**
  - Displays graphs for both real time and csv file temperature and conductivity data parsed in other classes

- **DepthLightActivity.java**
  - Displays graphs for both real time and csv file depth and light data parsed in other classes

- **ActivitySwipeDetector.java**
  - Detects when the user swipes left or right in empty space on any of the activities
  - Switches to whatever activity the user has swiped in the direction of

- **XML layout files**
  - There is a layout xml file for all of the activities in this application
    * These layout files contain the design specifications for each activity
    * They are manipulated programmatically in the corresponding activities
  - To do almost anything display related (that doesn't already exist in the application), you will need to first create the design/layout for your activity in an xml layout file
  - Next you will need to create an object that is linked to the xml file in your activity (the .java file)
  - From this point you can manipulate the display within the bounds of what is supported for each design by using methods from the object in the activity you are working in

5 Debugging and errors

- If the user gets an "Error getting file names, please try again" message multiple times in a row, the Arduino is most likely sending wrong data or is unable to read from the SD card. Alternatively, if this button is pushed before there is any data on the SD card (logfile has not been run yet) it will also display this message. **Try running the logfile command and recording a file (the download button above the SD card button).**

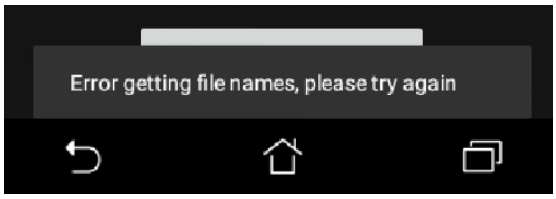


Figure 8: An error occurs when getting information about files on SD card

- If the user gets data that doesn't make sense in the SD card file display, the Arduino could be in the process of logging which needs to be stopped to fix this problem
- Clicking a correctly printed file name should still work even if there is bad data in the display

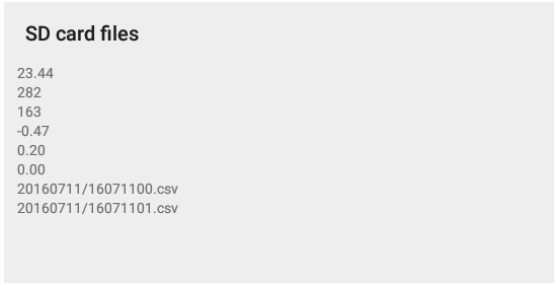


Figure 9: Nonsense data when Arduino is logging while sending us information about what's on the SD card. Clicking either of the correct file names at the bottom will still work.

6 Future improvements

6.1 Real time button syncing

- To make real time button syncing reliable on the Android side, a check is run in the syncButton() method of 3 of the classes. However, while this check protects the Android side from any issues, it does not protect the Arduino side.
  - The button syncing process will attempt to send a logapp command to the Arduino to stop data logging, but sometimes it does not properly detect the situation where it should stop the stream
  - This will not cause errors on either side, but logging may continue without the user's knowledge on the Arduino side

  - The alternate method that is very reliable for consistent button syncing is to just send a logapp every time the button is pressed. However, this method is vulnerable to the user mashing the button too quickly, sending it out of sync.
  - This will most likely be fixed by me, but if it is not then this is a great place to start working

6.2 Accelerometer data

- In the current version of the application, accelerometer data is being parsed and stored, but nothing is done with it
- The next developer of this application could easily add a display for this data on a new activity, or somewhere in the MainActivity

6.3 Better rotation

- In the current version of the application, to display displacement data calculated from gyroscope data, a 2D image of a seaperch is rotated in three dimensions
- In a future version of this app, a fully 3D image of a seaperch could be roated and animated

7 Licensing

- The Android application uses the Apache License, found at: `http://www.apache.org/licenses/LICENSE-2.0`