PHYSICS 3P92 - WINTER 2020

Experimental Physics II

Ryan Epp

Project Presentation

April 24, 2020

Objectives

- Bluetooth communication between a soil monitor and a mobile application.
- Standalone operation of the soil monitor, including battery operation.
- A lightweight app that doesn't require constant pairing/unpairing.

Requirements

Arduino:

- Battery operation for long periods of time.
- Soil readings around once a day.
- Timestamped soil readings.

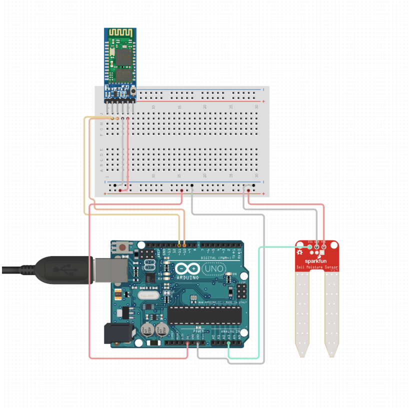
App

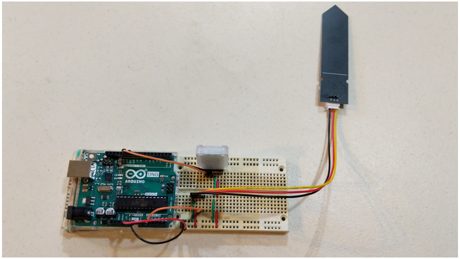
- Hassle free connection and data retrieval over Bluetooth.
- Clean and easy to read interface.

Components

- Arduino Uno
- Capacitive Soil Moisture Sensor v1.2
- DSD Tech HC-05 Bluetooth Module
- Android phone

Board Setup


UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN
sparkfun
Soil Moisture Sensor


UNO
ARDUINO

Board Details

Uno receives an analogue signal from the sensor and stores it while also recording the date/time.

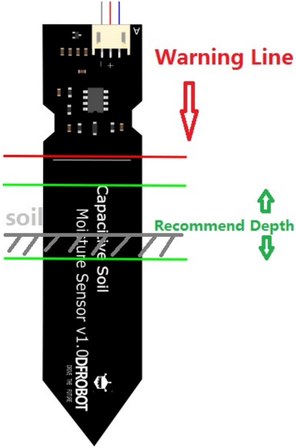
While running, it checks the BT read buffer and sends it's data back if needed.

Readings

Arduino takes analogue readings from 0 - 1000.

In air, the sensor reads back at around 580 - 620.

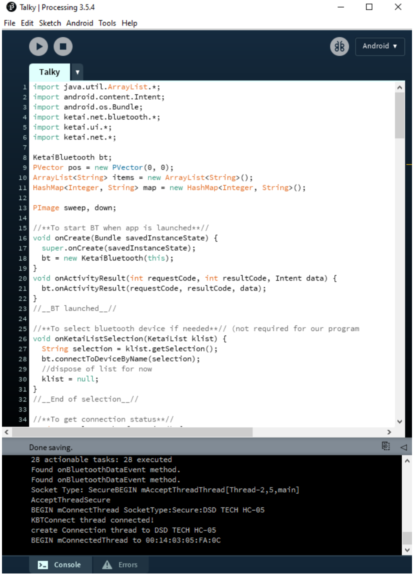
Less resistance gives a lower number.



Android App

The app itself was made with Processing (https://processing.org/), a flexible graphics development tool that uses Java and C.

The program builds the apk directly onto the Android device connected to the computer, you can also export it.

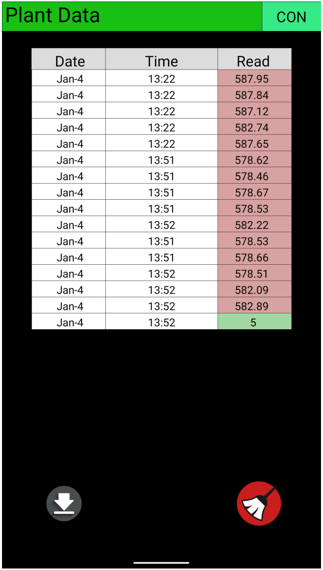


```
import java.util.ArrayList.*;
import android.content.Intent;
import android.os.Bundle;
import ketai.net.bluetooth.*;
import ketai.ui.*;
import ketai.net.*;

KetaiBluetooth bt;
PVector pos = new PVector(0, 0);
ArrayList<String> items = new ArrayList<String>();
HashMap<Integer, String> map = new HashMap<Integer, String>();

PImage sweep, down;

//**To start BT when app is launched**//
void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  bt = new KetaiBluetooth(this);
}
void onActivityResult(int requestCode, int resultCode, Intent data) {
  bt.onActivityResult(requestCode, resultCode, data);
}
//__BT launched__//

//**To select bluetooth device if needed**// (not required for our program
void onKetaiListSelection(KetaiList klist) {
  String selection = klist.getSelection();
  bt.connectToDeviceByName(selection);
  //dispose of list for now
  klist = null;
}
//__End of selection__//

//**To get connection status**//
```

Done saving.

```
28 actionable tasks: 28 executed
Found onBluetoothDataEvent method.
Found onBluetoothDataEvent method.
Socket Type: SecureBEGIN mAcceptThreadThread[Thread-2,5,main]
AcceptThreadSecure
BEGIN mConnectThread SocketType:Secure:DSD TECH HC-05
KBTConnect thread connected!
create Connection thread to DSD TECH HC-05
BEGIN mConnectedThread to 00:14:03:05:FA:0C
```

App details



Date	Time	Read
Jan-4	13:22	587.95
Jan-4	13:22	587.84
Jan-4	13:22	587.12
Jan-4	13:22	582.74
Jan-4	13:22	587.65
Jan-4	13:51	578.62
Jan-4	13:51	578.46
Jan-4	13:51	578.67
Jan-4	13:51	578.53
Jan-4	13:52	582.22
Jan-4	13:51	578.53
Jan-4	13:51	578.66
Jan-4	13:52	578.51
Jan-4	13:52	582.09
Jan-4	13:52	582.89
Jan-4	13:52	5

BT connection is done through MAC address (status in top right). It also has options to connect to a specific device name ("HC-05") if needed.

The two icons on the bottom ask for data again and clear the screen.

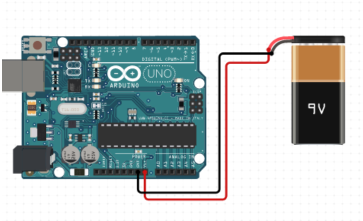
Because of the direct nature of the connection, the default password that is usually required isn't (allowing for new users to connect).

Complications

- Arduino Uno doesn't have an RTC, only the time/date the program started as a String
  - Arduino stores it's start date and time in Strings, just need to parse them into nubmers
- Bluetooth libraries for Processing are deprecated.
  - Processing *can* use the standard Android API (but it's not pretty).

Didn't go as Planned

- Missing materials for battery operation
- Time parsing is off for the board.
- App could've looked better.
- App usually doesn't get some data.


UNO
ARDUINO
9V