Advanced Computer Systems
Ben Sterenson
Michael Palarya

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Tel-Aviv University
Prof. Sivan Toledo

PlantIO - Documentation

Description

- PlantIO is a smart IoT plant system that specializes in monitoring plants by sampling Soil Moisture, Temperature and Light levels.
- PlantIO was developed as part of Advanced Computer Systems Course.
- In order to simulate a complete system we use a combination of real sensor readings from a Soil Moisture sensor, board's temperature and randomly generated light levels.

Solving The Problem

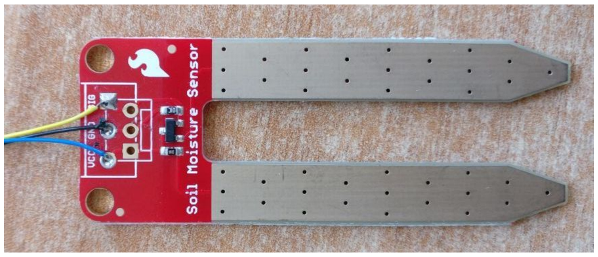
- Urban planting is getting more and more popular, many people grow spices and ornamental plants in their homes. Although the popularity, many don't know how, or find it hard to remember to water the plant and give it the right amount of water in the right time.
- PlantIO is trying to solve this problem by monitoring each plant's Soil Moisture, Temperature and Light exposure, giving the user a full picture of the plant, making the growing process easy and accurate.
- PlantIO is meant for any house that wants to grow some plants and get a better picture on the process of planting. Most of the smart solutions today are very complicated, expensive and energy-consuming.
- By using PlantIO the user does not waste water and can give the optimal amount that is needed to grow the plant in the best way. Furthermore, it is cheap and very energy-efficient way to monitor plants without the need to buy fancy robotic and computer systems.

Equipment:

1. SparkFun Soil Moisture Sensor - https://www.sparkfun.com/products/13322
2. SimpleLink™ CC2650 Wireless MCU LaunchPad - http://www.ti.com/tool/launchxl-cc2650
3. Hook-Up Wire - Assortment https://www.sparkfun.com/products/11375
4. Break Away Headers - Straight https://www.sparkfun.com/products/116

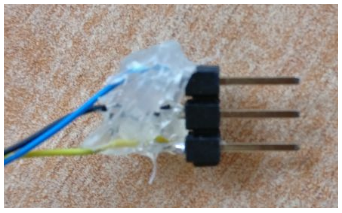
Hookup Guide:

1. Solder some wire to the sensor as following:

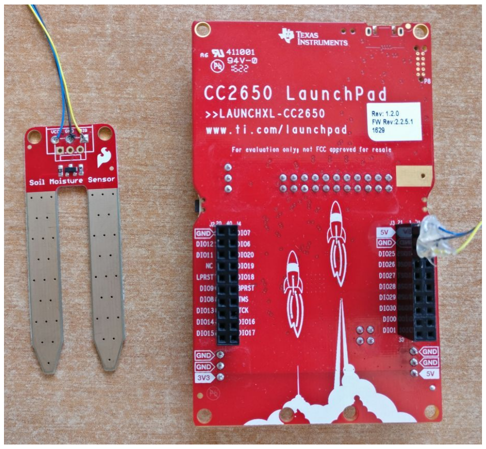


- Use colors to distinguish between Voltage, Ground and Signal wires.

2. Cut 3 pins from the Break Away Headers and Solder the wire to them.



3. Plug the 3 pins to (5v, GND and DIO25 ports) Your final board should look as following:



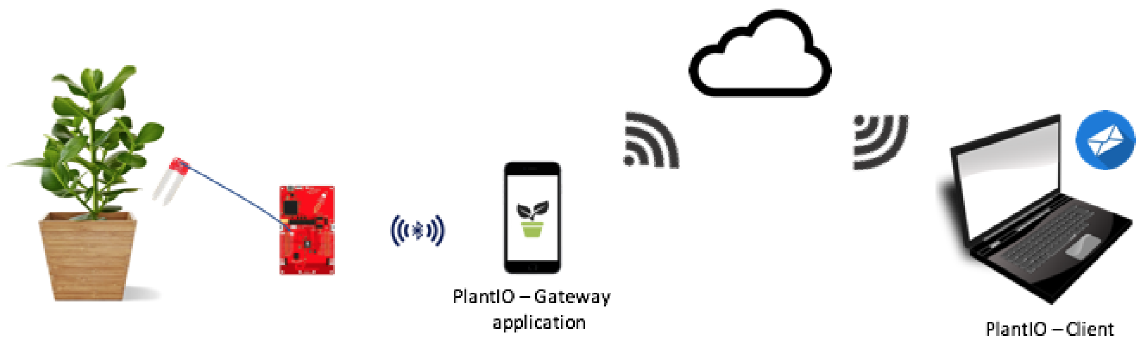
Software Setup:

1. Code Composer Studio - http://www.ti.com/tool/ccstudio
2. TI BLE Stack v2.2.1 - http://www.ti.com/tool/BLE-STACK-ARCHIVE
3. Visual Studio 2017 - Xamarin https://www.visualstudio.com
4. Python 2.7.X - https://www.python.org/
5. Python Matplotlib library - https://matplotlib.org/
6. Python Requests library - http://docs.python-requests.org/en/master/
7. Using google drive - Create a google sheet with the following column names:

A	B	C	D	E	F	G
id	date	time	type	scale	value	ble_sample_type

8. Copy sheet Url to https://sheetsu.com
9. Copy API URL https://sheetsu.com/apis/v1.0/YOUR_API_ID and Enter API URL to PlantIO app.
10. Download project source files from https://github.com/BenSterenson/PlantIO

High Level System Illustration:



From left to right:

- Sensors are inserted into the plant's soil to sample the data
- The board transfers the measured data over a BLE protocol to a mobile phone
- The phone sends the data to a cloud-based database over a WiFi connection
- Data is provided from the cloud to the client over a WiFi connection to extract the information needed in order to display statistics and graphs

Modules - Overview

Texas Instruments CC2650 (Hardware):

- Responsible for collecting data from Soil Moisture Sensor and board temperature (board temperature simulates the temperature the plant is exposed to).
- Normalizes data into predetermined human-readable scale.
- Sample and publish data via BLE - Supports periodic read/subscription to notification.
- Subscription to notification - Once board identify 10 difference from last sample it will publish it to mobile client.

Mobile Application (Gateway):

- Used as a **gateway** to transfer measurements from CC2650 board to cloud.
- Simple user interface supports periodic read and auto notification.
- Sends real time data to cloud.

Google Drive (Cloud):

- Responsible for storing measurements received from PlantIO application.
- Interaction with google sheet by REST API.
- Google sheet fields - **id, date, time, type, scale, value, ble_sample_type**

Python Graph Plotting App (Client):

- Responsible for extracting data and representing a graph with measurements.
- Send mail with attached graph.

Interfaces - Overview

Hardware - Gateway:

- Data is sent from Hardware to the Gateway using BLE Notification on every state change.
- Data is requested from Hardware by the Gateway using a BLE Read Request.

Gateway - Cloud:

- Data is uploaded through the Gateway to the Cloud over WiFi using REST API.

Cloud - Client:

- Data is downloaded from the Cloud by the Client over WiFi using REST API.

Texas Instruments CC2650 (Hardware)

Software

The software that is running on the TI CC2650 board was written using Code Composer to run on TI-RTOS real time operating systems.
We based our code on the Project Zero by TI and made the needed adjustments to add the necessary logic for it to behave as expected.

- The planning and design of the project intended to focus on matters discussed through the course such as Power Efficiency, Tasks, HW/SW-context and semaphores under the restrictions of the operating system, and the limited onboard memory.

Operation

The Program's name is PlantIO and it begins by advertising itself over BLE protocol.

Using TI's manuals we defined two new modules, the first module samples the sensors and presents them in a human-readable way, and the second module is a new BLE service that reports this data to a listening device.

Once connected it starts to sample the sensors, Soil Moisture and Temperature, periodically every 15 minutes by default.

We thought about making the sampling rate configurable but decided that this is an unneeded ability, and 4 samples per hour gives us all needed data without spending too much energy.

On meaningful changes - defined as 10% loss of water, or 1C degree change in temperature, the board sends a notification to the listening client with the current measurement.

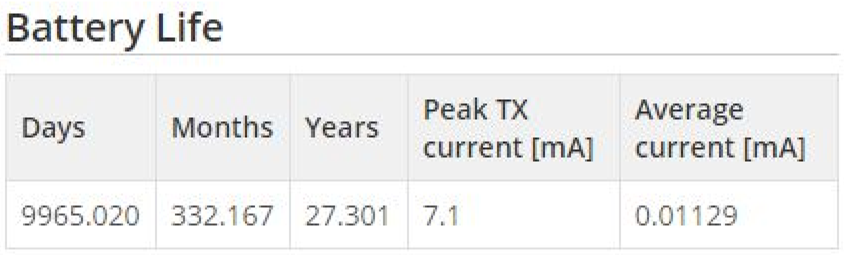
During the regular session, the board will respond to any Read Request directed to it from the Client with all data collected on last measure.

Power Consumption

Our CC2650 LaunchPad kit did not operate well under two AA alkaline batteries with a total voltage of 3V. The system became stable under 3.3V supply, and all calculations were made assuming the usage of two AA alkaline batteries of 2600mAh supplying a total of 3.3V.

In order to estimate power consumption caused by BLE communication, Texas Instruments' BLE power consumption calculator came in handy:
http://www.ti.com/lsds/ti/wireless-connectivity/bluetooth-low-energy/power-calculator.page#

After entering all necessary parameters, the estimation is as follows:


Battery Life

Days	Months	Years	Peak TX current [mA]	Average current [mA]
9965.020	332.167	27.301	7.1	0.01129

BLE power consumption is negligible compared to the board's idle state and sensor sampling. When connecting the entire circuit together with the sensors, an oscilloscope measures a consumption of approximately ~64mA on average.

5200mAh divided by 64mA will grant us approximately 81 hours of operation, which are around three and a half days.

Taking into account the fact that our product is meant to be integrated in greenhouses under direct sunlight or an artificial equivalent, a simple solar panel that produces 1kWh a day on average (under 8 hours of sun) will suffice to power around 15 boards simultaneously.

Software Design

<u>Main Program</u>

The main program is running an infinite loop that checks for messages accumulating in a queue. Whenever a new message is enqueued it is handled according to its type.

<u>Sensor Management</u>
We start the initialization by calling init_sensors that is responsible for
1. Semaphore initialization
2. Task initialization - initialization of task that is responsible for sensor reading.

We define two global unsigned integers that holds the last sent, and recently measured data from the sensors -

```
// global int variable
uint16_t globalValue = 0x0000;
uint16_t globalOldValue = 0xFFFF;
```

A single variable holds data for both sensors, Soil Moisture - MSByte, and Temperature - LSByte. This saves us some memory and data sending operations on the expense of efficient bitwise operations.

Whenever a sensor read is requested (by BLE services for example) to sample for data, it updates the semaphore to perform the actual measure, that eventually sets the value in the global variable and it is further handled by the BLE services.

<u>Periodic Sampling of Sensors</u>
For the periodic sampling of sensors we defined a SW-context function to be called every time a clock timer is triggered.
The clock is initialized in the main initialization function, and it is activated on a button press.

Because a SW-context function may block the TI-RTOS operating system, we cannot call BLE API commands, so instead we define a unique message that is handled by a Task, and enqueue it using -

```
user_enqueueRawAppMsg(APP_MSG_PERIODIC_TIMER, NULL, 0)
```

- Note that is is not sending any data, just signaling the Task, so no blocking caused and the program keeps on running.

<u>BluetoothLE Services</u>
We defined a new BLE service and characteristics that support Read and Notify events.

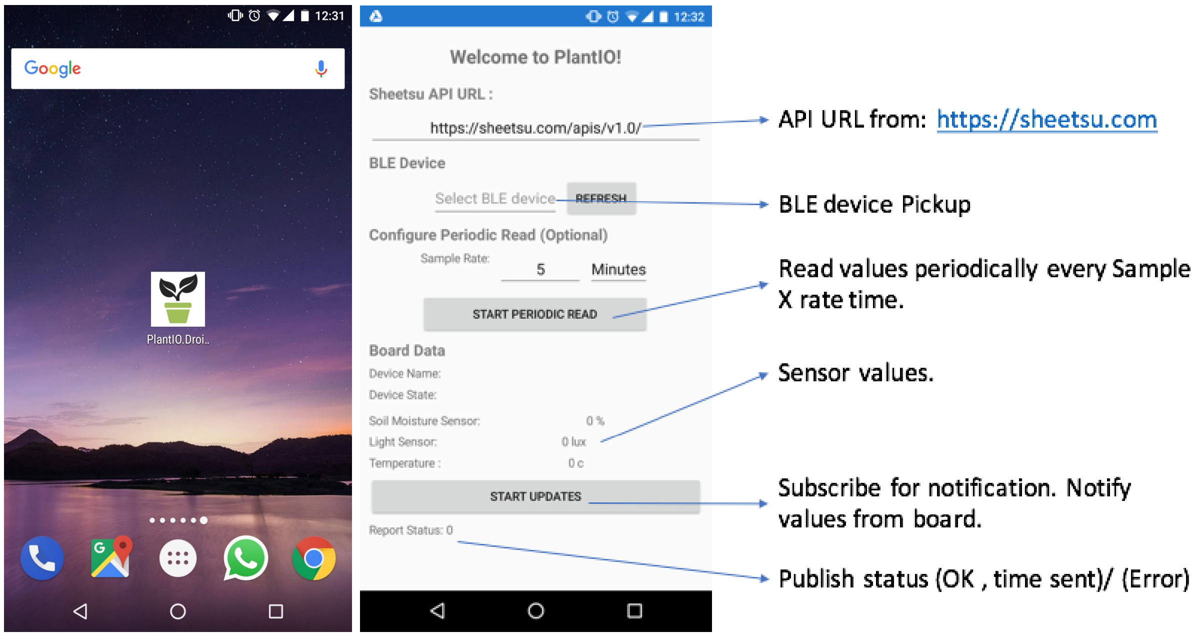
Whenever a Read Request arrives it is handled by the service DispatchHandler. The handler is a loval handler that is called from the Task context of the main Task that handles the different events in the main function.

Whenever the service receives new data, after a sensor reading, the ValueChanged CallBack Function is called.
The CB function asks the service what the received data was, and it sends it in a message to the user Task for processing.

Mobile Application (Gateway)

The mobile application was developed using **Xamarin** technology to offer a cross-platform application that is written once and ready to be compiled and run on any common operating system including iOS and Android.

PlantIO application is a one-page app -



Welcome to PlantIO!

Sheetsu API URL : https://sheetsu.com/apis/v1.0/

BLE Device: Select BLE device [REFRESH]

Configure Periodic Read (Optional)
Sample Rate: 5 Minutes
[START PERIODIC READ]

Board Data
Device Name:
Device State:
Soil Moisture Sensor: 0 %
Light Sensor: 0 lux
Temperature : 0 c
[START UPDATES]
Report Status: 0

API URL from: https://sheetsu.com

BLE device Pickup

Read values periodically every Sample X rate time.

Sensor values.

Subscribe for notification. Notify values from board.

Publish status (OK , time sent)/ (Error)

The application supports the following modes that work independently:

- Periodic Read:
  Sends a Read Request to recieve Soil Moisture and Temperature values from board.
  The requests are sent periodically in a user configurable rate - under “Sample Rate” field

- Notifications:
  Automatic Notifications are sent by the board, informing of critical Soil Moisture and Temperature changes.
  Critical changes are defined as 10% change in Moisture, or 1C degree in Temperature.

- In both cases, the collected data is transferred to the cloud using REST API, indicating device ID, time of update, sensor, the measurement, and mode used to send the data.

PlantIO Graph Plotting Application (Client)

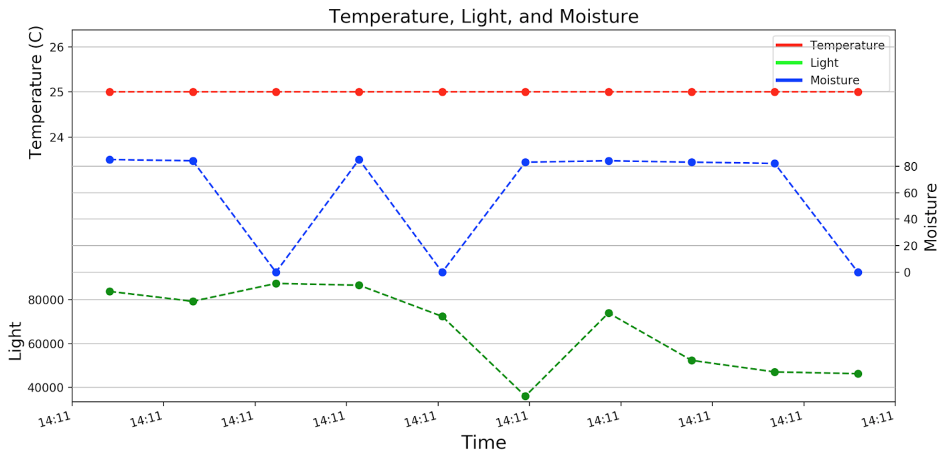
The PlantIO Graph Plotting Application was developed using **Python** and offers a cross-platform application that runs on Windows, OSX, and Linux.
The application generates a graph named "output.png" in PythonClient folder and sends a warning email to given email.

In order to run the PlantIO Graph Plotting go to PythonClient folder.

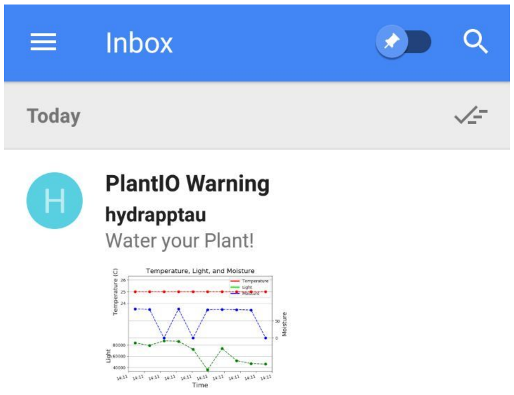
1. Installed required libraries (see Software Setup section)
2. Enable Less secure apps on Gmail - https://myaccount.google.com/u/0/lesssecureapps
3. Add a credentials file, the file should contain the following format "user:password"
4. Run the following command:

```
python PlantIO.py -c CREDENTIALS_PATH [-f FROM_DATE -t TO_DATE] -to YOUR@EMAIL.com
```

The graph shows three sub-graphs of Temperature, Soil Moisture and Light.
The application gives the user an easy way to monitor the plant and plot the data on a graph.



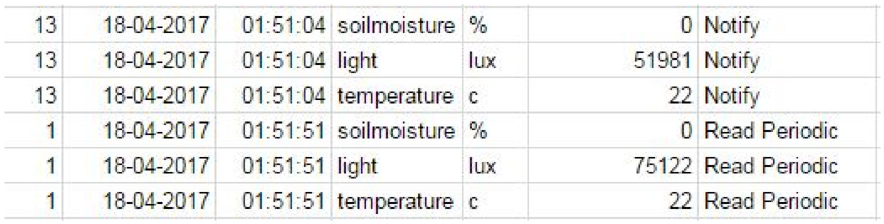
Furthermore the application is sending the graph by email to the selected address



PlantIO Warning
hydrapptau
Water your Plant!

Google Drive (Cloud)

The data is stored in and retrieved from the Google Drive Cloud-based database using Sheetsu REST API.
The data is stored in the following human-readable format:



13	18-04-2017	01:51:04	soilmoisture	%	0	Notify
13	18-04-2017	01:51:04	light	lux	51981	Notify
13	18-04-2017	01:51:04	temperature	c	22	Notify
1	18-04-2017	01:51:51	soilmoisture	%	0	Read Periodic
1	18-04-2017	01:51:51	light	lux	75122	Read Periodic
1	18-04-2017	01:51:51	temperature	c	22	Read Periodic

Message Protocol

1. **CC2650 TO PlantIO Mobile Application**
   PlantIO Application Mobile expects to receive from TI CC2650 a string of the following format "XX XX" where "X" represents a **HEX** number s.t. the first substring to the white space represents the temperature and the second represents the soil Moisture.

2. **PlantIO Mobile Application TO Google Sheets**
   PlantIO Application Mobile transfers data received using REST API.
   String format:

   "https://sheetsu.com/apis/v1.0/{YOUR_API_ID}/id/{SAMPLE_ID}/date/{%dd-%mm-%YYYY}/time/{%H:%M:%S}/type/{soilmoisture/light/temperature}/scale/{%/lux/c}/value/{VALUE}/ble_sample_type/{Notify/Read Periodic}"

3. **Google Sheets TO PlantIO graph plotting application**
   PlantIO Graph plotting Application makes use of REST API to read data from Google sheets.
   String format:

   "https://sheetsu.com/apis/v1.0/{YOUR_API_ID}"

Future Development and Plans

1. Adding support for temperature, light and other sensors that can give a larger picture of the plant growing.

2. Adding support for a sets of sensors such that a single board will monitor multiple plants.

3. Create a network of boards that can all communicate with each other and transfer data from one board to another via BLE until the data reach the gateway.

4. Adding support to store data on board’s flash drive.

5. Make Mobile application support multiple boards in parallel.

6. Creating a task that auto-generates graph pre-defined time frame.

7. Analyzing data and building a statistical data set to implement Machine Learning algorithms to better monitor different types of plants.