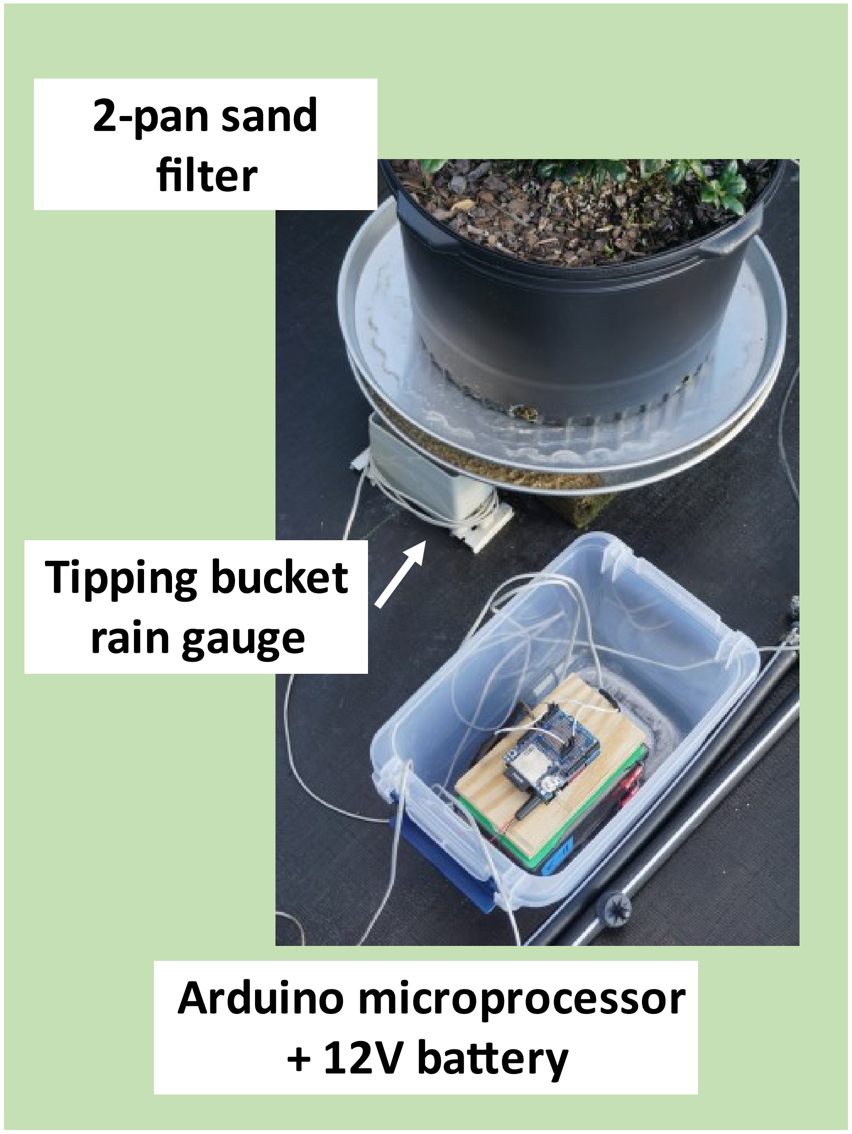
# A Guide for Using the WaterTips App to Monitor and Adjust Irrigation in Container Nurseries

**WaterTips** is a mobile device app, which along with tipping-bucket rain gauge sensors and battery-powered microprocessors, allows producers to monitor leachate (drainage) for guiding irrigation in container nurseries. A protype set up is depicted below. The Arduino microprocessor communicates via Bluetooth with the WaterTips application.


2-pan sand filter
Tipping bucket rain gauge
Arduino microprocessor + 12V battery

## Leachate fraction (LF)

**Why is leachate monitored?**

The amount of leachate is an indicator of irrigation efficiency. If leachate volume is consistently low, water deficits in the container substrate can lead to water insufficiency and reduced growth. If leachate volume is high, excessive water is being applied, which wastes water, consumes energy, and hastens the leaching of applied chemicals including fertilizer.

**What is the leaching fraction and how is it used to guide irrigation?**

The leachate fraction (LF) is the amount of leachate (or container drainage) relative to the amount of irrigation water applied to a container:

**LF = leachate/water applied * 100%**

When monitored routinely, irrigation can be adjusted to target LF values of 15-25%. An equation to adjust irrigation is:

**New run time = (100%-LF) ÷ (100%-target LF) × Previous run time**

**How is leachate measured and what role does WaterTips play?**

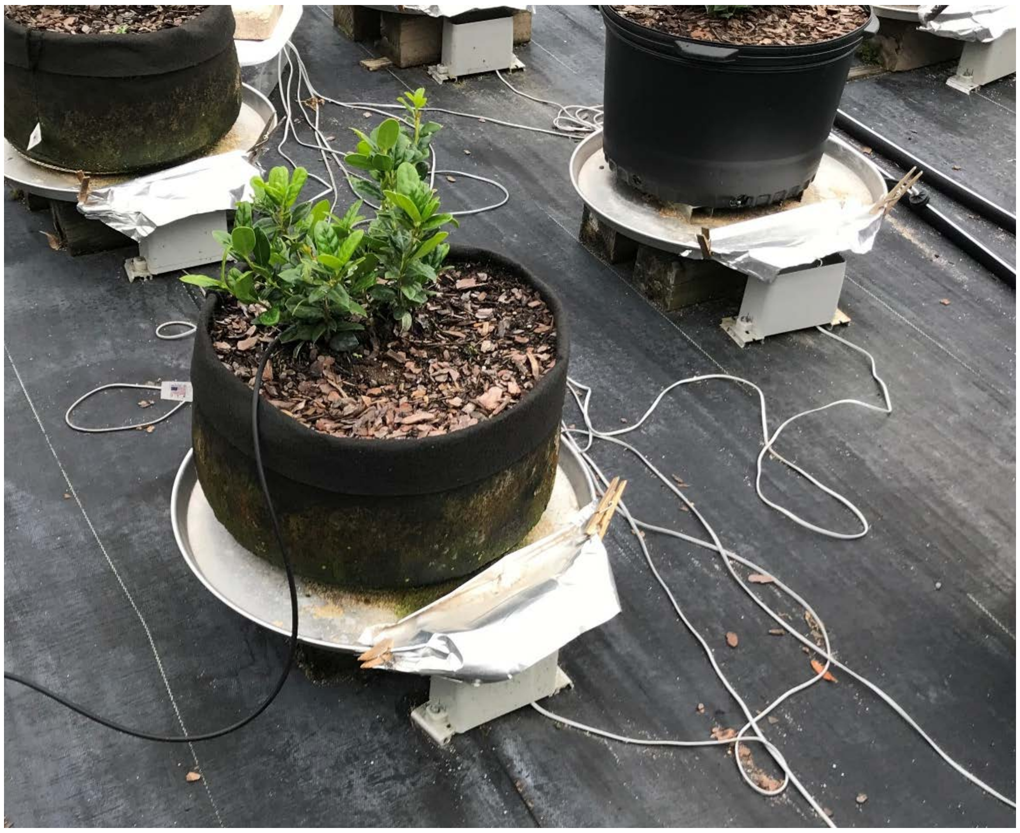
Leachate from containers is directed into tipping bucket rain gauges using aluminum pans with drain holes. A battery-powered microprocessor (Arduino) wired to the rain gauges, records the daily number of tips from each rain gauge (3-4 rain gauges per irrigation zone). WaterTips, running on a mobile phone, acquires the daily tip data from the Arduino via a Bluetooth connection. WaterTips calculates the average leachate volume from the 3-4 rain gauges using a tip volume of 1.7 mL/tip. WaterTips also estimates the amount of water applied to the container based on the irrigation run time and irrigation rate. From the average leachate volume and the volume of irrigation water applied, WaterTips calculates the LF and an adjusted or desired irrigation run time using the equations above. The user can save LF data providing a history of LF and irrigation run times.

## What equipment is needed besides the WaterTips app?

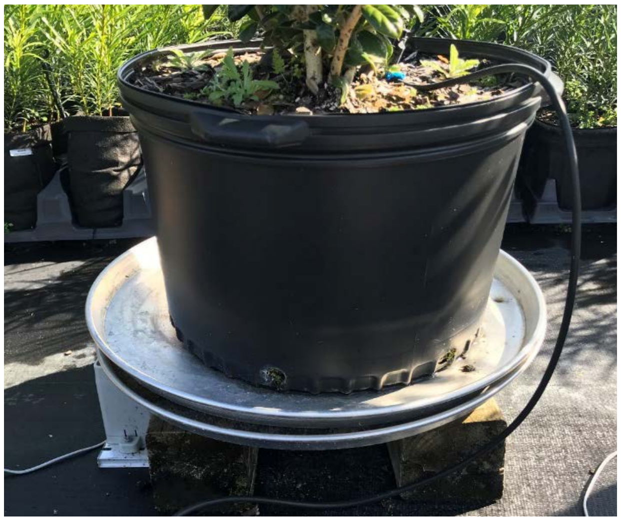
The main components include leachate collection setups, tipping-bucket rain gauges, and a Bluetooth-enabled Arduino (Bluno) microprocessor for logging tipping bucket data. A list of equipment/materials is given in a later section.

**Leachate collection**

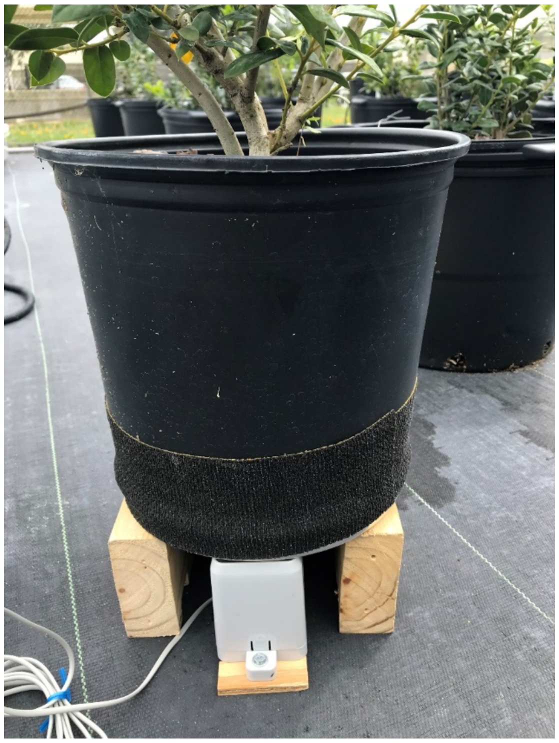
Containers are placed on an elevated aluminum pizza pan to collect leachate and direct the leachate via a drain hole into a tipping bucket rain gauge placed underneath the drain hole. Rain gauges are designed for measuring rain, not spray-stake irrigation, which applies water at a much higher rate (>10X) than sprinkler irrigation. Therefore, for spray-stake microirrigation, a modification is needed to slow the leachate flow into the rain gauge.  We have used two methods for this both entailing a sand filter.  For sprinkler irrigation, which applies water at a similar rate as rain, no sand filter is needed.



***One-pan method***   The diameter of the pizza pan is ≈4 inches wider than the bottom diameter of the container. This allows the placement of a sand filter bed at the low end of the pan, above the drain hole. To retain the sand, a hose washer screen is attached under the drain hole with silicone adhesive sealant. If leachate volume is large, leachate can pool and re-absorption by the container is possible. The larger the pan, therefore, the larger the reservoir for holding leachate without re-absorption. Because the sand filter is exposed, it can be covered with a piece of aluminum foil to limit evaporation and keep out debris. Ideally, the sand filter remains saturated so that leachate volume collected is not affected by the sand filter.

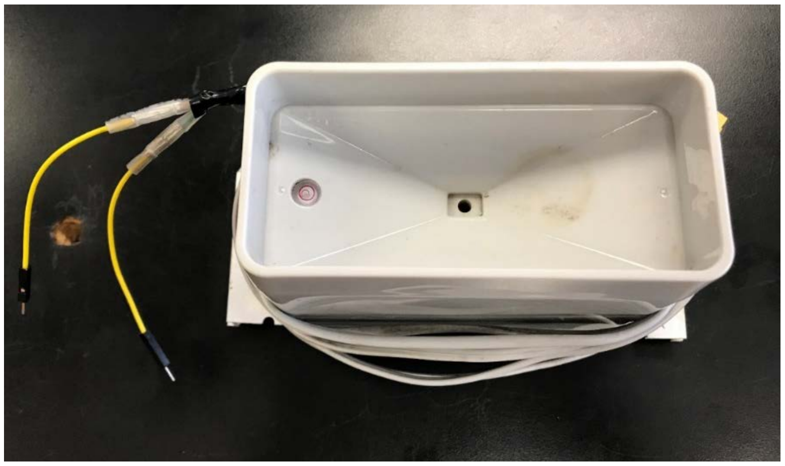


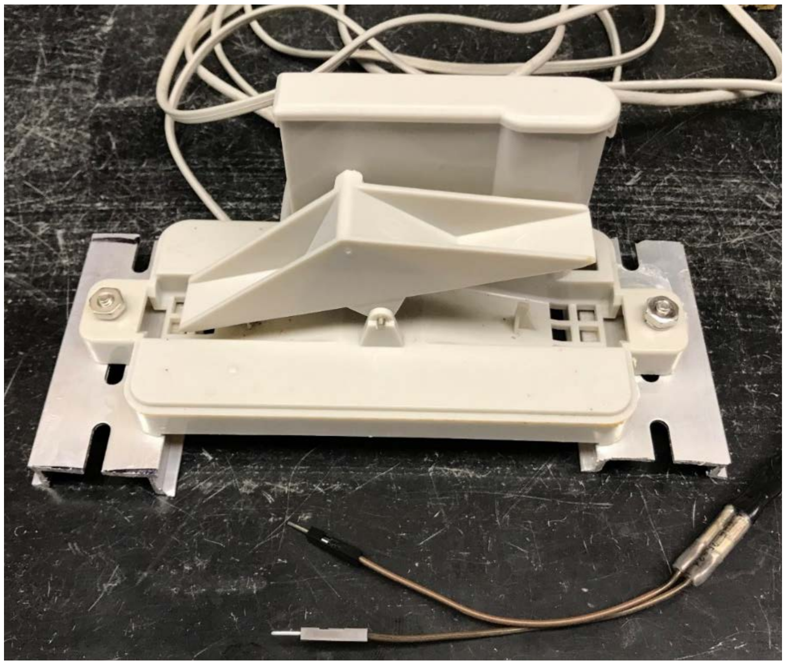
***Two-pan method*** The bottom pan with sand filter is setup to drain into the rain gauge as in the one-pan method. The same-sized top pan is angled to drain in the opposite direction using 2”x2” small ½”-thick plywood support pieces strategically placed on bottom pan. The top pan limits evaporation from sand filter, keeps out debris, and provides a greater volume for storing leachate without re-absorption by container.



***One-pan method for sprinkler*** Tape or sports wrap can be used to prevent sprinkler irrigation water from directly entering the collection pan that drains into the rain gauge.

## Tipping bucket rain gauge

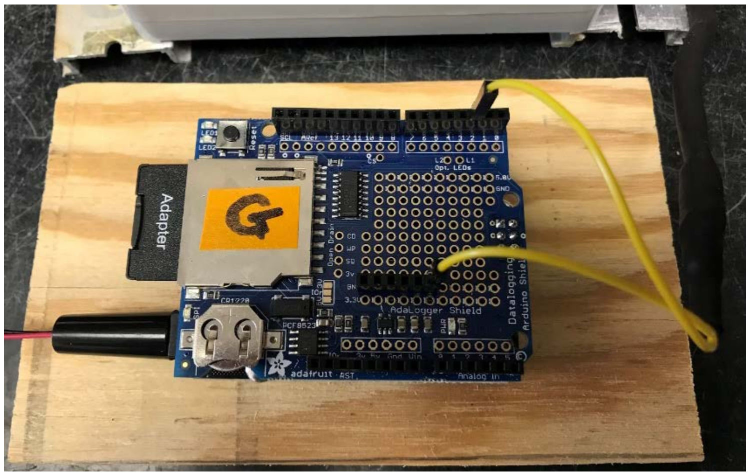




The tipping bucket rain gauge has two buckets or compartments that pivot on an axle. When a bucket fills and tips, a magnet on the bucket mechanism activates a reed switch that allows a pulse of current to flow to the Arduino. After one bucket tips and empties, the other bucket begins filling. Calibrating the bucket provides a volume per tip (1.7 mL/tip) that can be multiplied by the number of tips to provide a leachate volume.

## Arduino microprocessor

A battery-powered Arduino microprocessor with a data-logging shield is used to count pulses (tips) from 1-4 tipping bucket rain gauges. In the photo below, only one rain gauge is connected. The Arduino provides Bluetooth connectivity for uploading data to WaterTips.

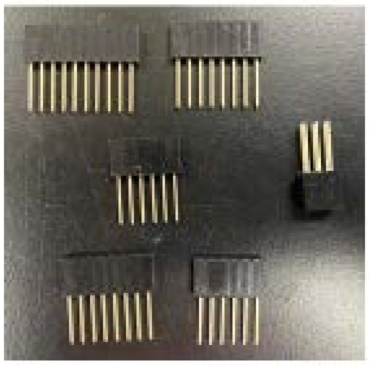


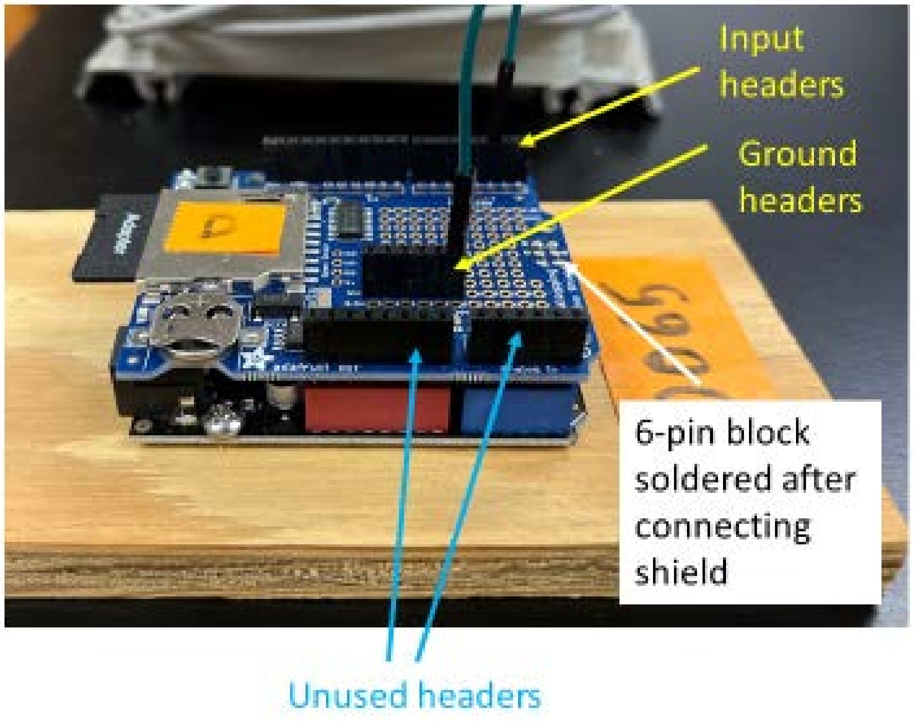
**Assembling the Arduino**

The data-logging shield or top piece is attached to the Arduino with header pins bought separately (photo below). The header pins come in blocks that fit the Arduino configuration. An additional block of 6 header pins will also need to be soldered to the Arduino ground circuit. Before soldering, the pins are first placed on the data shield so that the pins protrude or face down - the receiving ends will face up. Turning the data shield over, while maintaining the pins in the correct positions, allows the protruding pins to be soldered to the shield. Once soldered, the shield is turned back over so that the protruding pins are facing down and receiving ends facing up. Lastly, the data-logging shield comes with a 6-pin (2x3) header that once positioned on the Arduino, the protruding ends will protrude up through the data-logging shield once shield is attached to the Arduino. The final assembly is to press the pins into the Arduino and then solder the 6-pin leads from the Arduino that protrude up through the shield.

A tipping gauge is connected to the Arduino by inserting one jumper from a gauge into one of four available input header pins:  2, 3, 6, and 9. The other jumper pin from the gauge is inserted into any of the six ground header pins.

Header pins before soldering





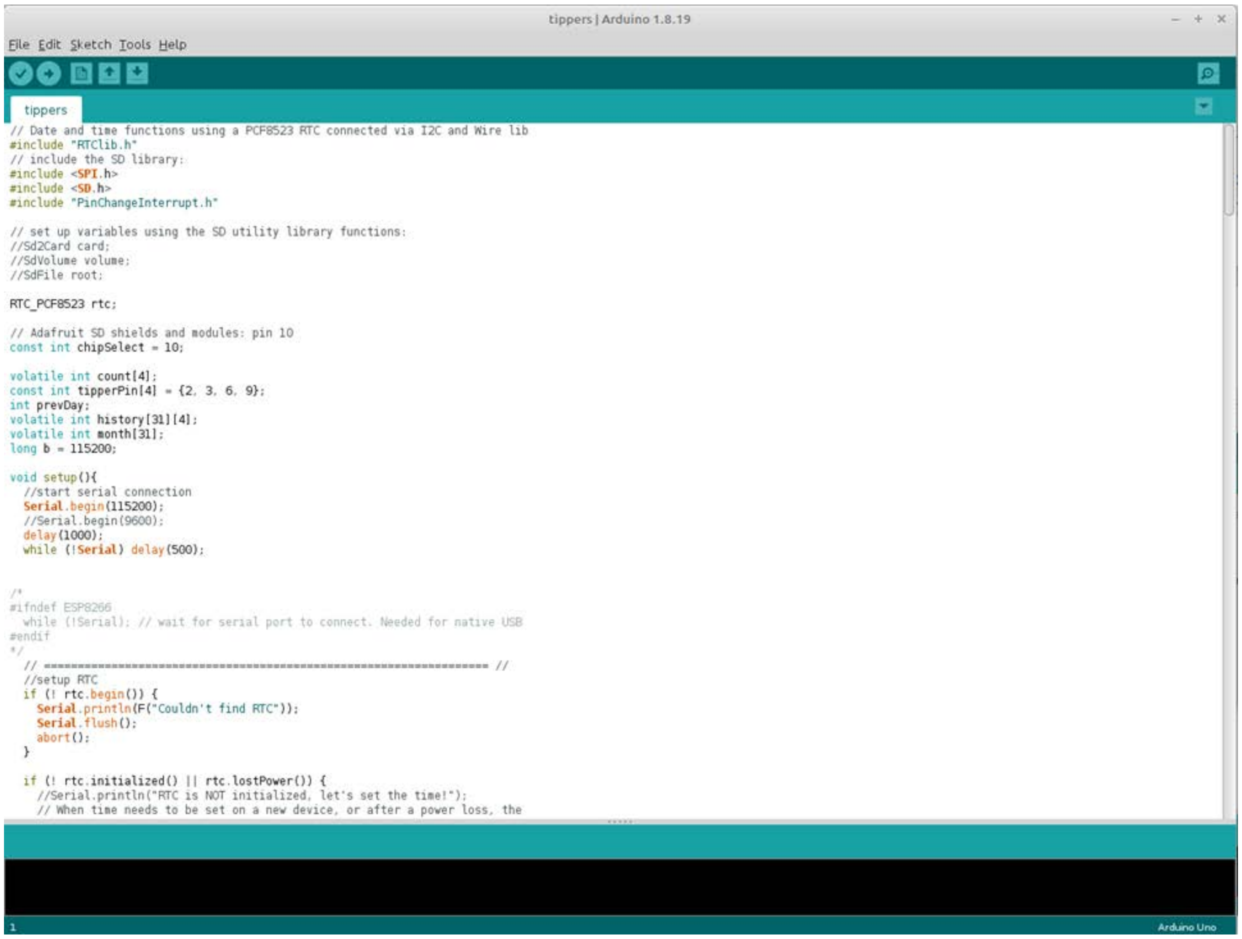
**Programming the Arduino**

Follow these steps to program the Arduino:

1. The first thing needed is to set up the Arduino IDE software. Go to the Arduino homepage https://www.arduino.cc/en/Guide and follow the Getting Started Guide to download and install the Arduino Desktop IDE. The Arduino Desktop IDE can be installed on any Linux, Mac, or Windows computer.

2. Make a folder named tippers and copy the tippers.ino (Arduino program) in this folder. Note that the tippers.ino file has to be in a folder with the same name as the program (minus the .ino).

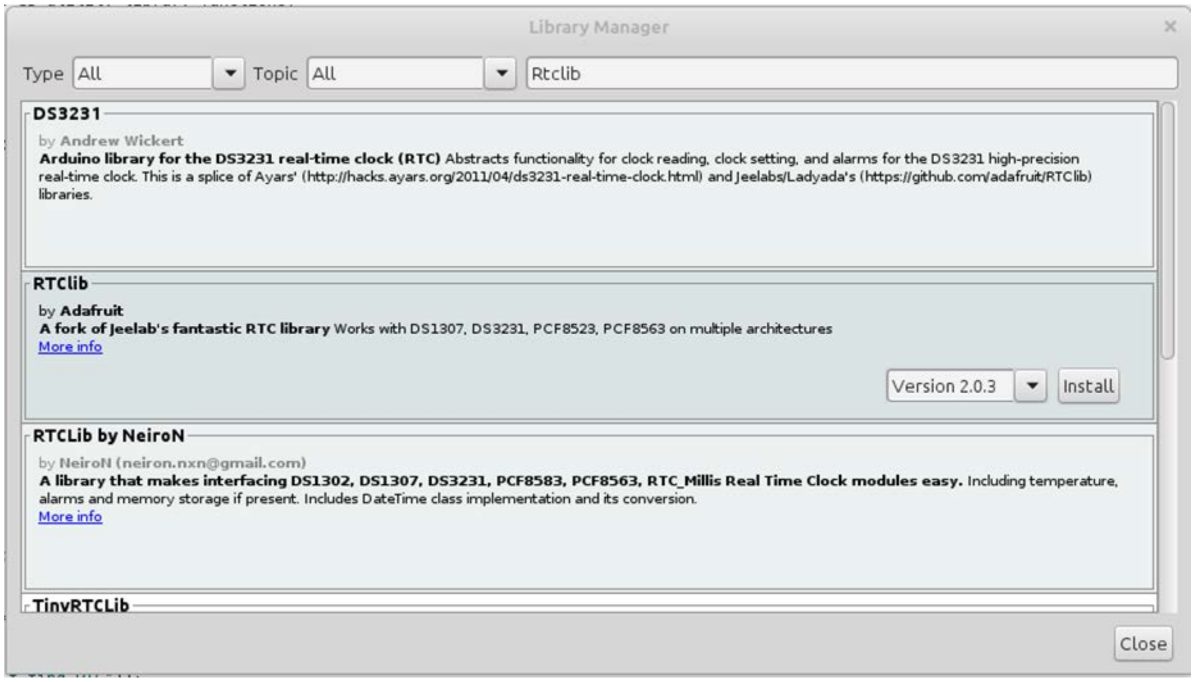
3. Open the Arduino IDE. Select File – Open and then navigate to the tipper.ino file and open it.



```
// Date and time functions using a PCF8523 RTC connected via I2C and Wire lib
#include "RTClib.h"
// include the SD library:
#include <SPI.h>
#include <SD.h>
#include "PinChangeInterrupt.h"

// set up variables using the SD utility library functions:
//Sd2Card card;
//SdVolume volume;
//SdFile root;

RTC_PCF8523 rtc;

// Adafruit SD shields and modules: pin 10
const int chipSelect = 10;

volatile int count[4];
const int tipperPin[4] = {2, 3, 6, 9};
int prevDay;
volatile int history[31][4];
volatile int month[31];
long b = 115200;

void setup(){
  //start serial connection
  Serial.begin(115200);
  //Serial.begin(9600);
  delay(1000);
  while (!Serial) delay(500);


/*
#ifndef ESP8266
  while (!Serial); // wait for serial port to connect. Needed for native USB
#endif
*/
  // ================================================================= //
  //setup RTC
  if (! rtc.begin()) {
    Serial.println(F("Couldn't find RTC"));
    Serial.flush();
    abort();
  }

  if (! rtc.initialized() || rtc.lostPower()) {
    //Serial.println("RTC is NOT initialized, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
```

4. The first time you install the Arduino IDE ONLY, you will need to download a couple of libraries:

a) from the top navigation menu, select Tools- Manage Libraries and the Library Manager will pop up.



**DS3231**
by Andrew Wickert
**Arduino library for the DS3231 real-time clock (RTC)** Abstracts functionality for clock reading, clock setting, and alarms for the DS3231 high-precision real-time clock. This is a splice of Ayars' (http://hacks.ayars.org/2011/04/ds3231-real-time-clock.html) and Jeelabs/Ladyada's (https://github.com/adafruit/RTClib) libraries.

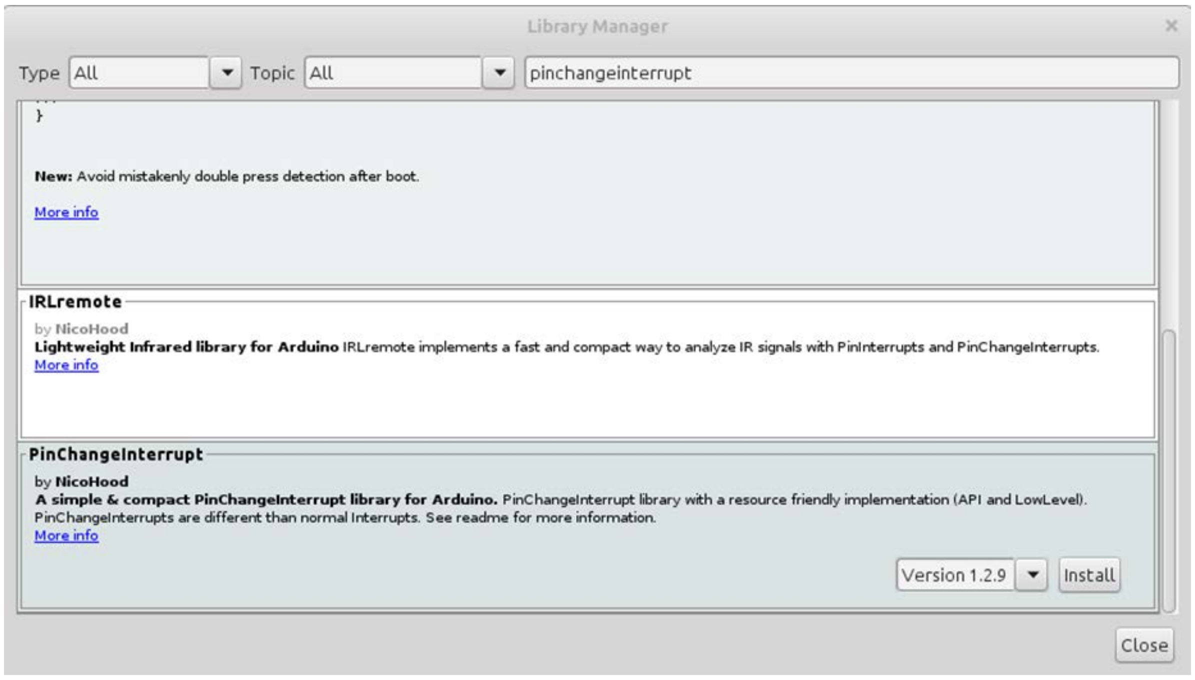
**RTClib**
by **Adafruit**
**A fork of Jeelab's fantastic RTC library** Works with DS1307, DS3231, PCF8523, PCF8563 on multiple architectures
More info

**RTCLib by NeiroN**
by NeiroN (neiron.nxn@gmail.com)
**A library that makes interfacing DS1302, DS1307, DS3231, PCF8583, PCF8563, RTC_Millis Real Time Clock modules easy.** Including temperature, alarms and memory storage if present. Includes DateTime class implementation and its conversion.
More info

**TinyRTCLib**

b) In the box labeled Filter you search, type **Rtclib.** You will see a library called RTClib by Adafruit. Mouse over it and you will see an install button. Click this. It may pop up a dialog box saying that the library **RTClib** needs some other library dependencies not currently installed. If so, click **Install all**.

c) Repeat this process to install **PinChangeInterrupt** by NicoHood. After these two libraries are installed, you may close the Library Manager.



}

**New:** Avoid mistakenly double press detection after boot.

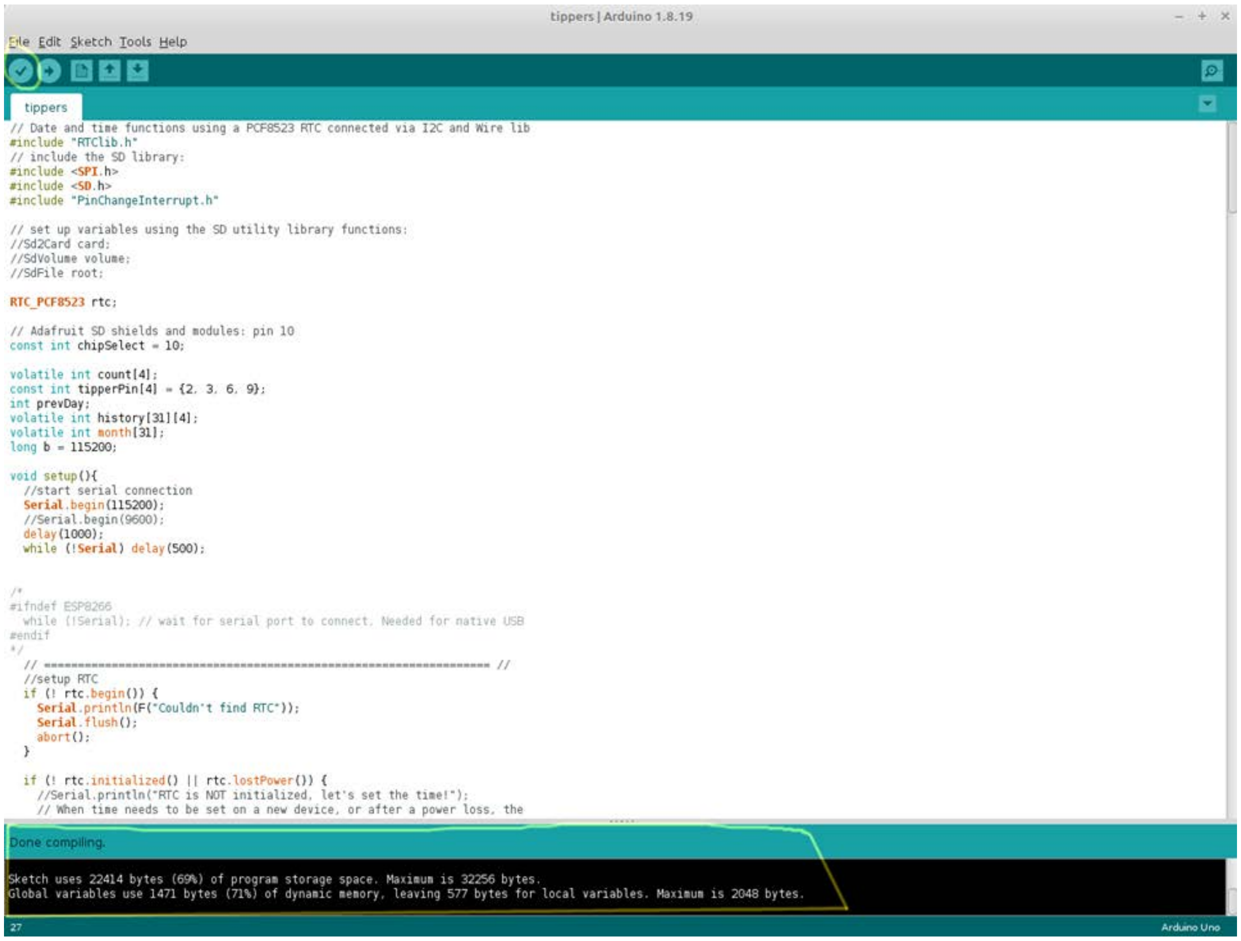
More info

**IRLremote**
by NicoHood
**Lightweight Infrared library for Arduino** IRLremote implements a fast and compact way to analyze IR signals with PinInterrupts and PinChangeInterrupts.
More info

**PinChangeInterrupt**
by **NicoHood**
**A simple & compact PinChangeInterrupt library for Arduino.** PinChangeInterrupt library with a resource friendly implementation (API and LowLevel). PinChangeInterrupts are different than normal Interrupts. See readme for more information.
More info

5. Install the Tippers.ino program.

a) After the Arduino IDE has been installed, simply start it and use the top menu bar to select File – Open and browse to the tippers.ino file and open it.



```
// Date and time functions using a PCF8523 RTC connected via I2C and Wire lib
#include "RTClib.h"
// include the SD library:
#include <SPI.h>
#include <SD.h>
#include "PinChangeInterrupt.h"

// set up variables using the SD utility library functions:
//Sd2Card card;
//SdVolume volume;
//SdFile root;

RTC_PCF8523 rtc;

// Adafruit SD shields and modules: pin 10
const int chipSelect = 10;

volatile int count[4];
const int tipperPin[4] = {2, 3, 6, 9};
int prevDay;
volatile int history[31][4];
volatile int month[31];
long b = 115200;

void setup(){
  //start serial connection
  Serial.begin(115200);
  //Serial.begin(9600);
  delay(1000);
  while (!Serial) delay(500);


/*
#ifndef ESP8266
  while (!Serial); // wait for serial port to connect. Needed for native USB
#endif
*/
  // ==================================================================== //
  //setup RTC
  if (! rtc.begin()) {
    Serial.println(F("Couldn't find RTC"));
    Serial.flush();
    abort();
  }

  if (! rtc.initialized() || rtc.lostPower()) {
    //Serial.println("RTC is NOT initialized, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
```

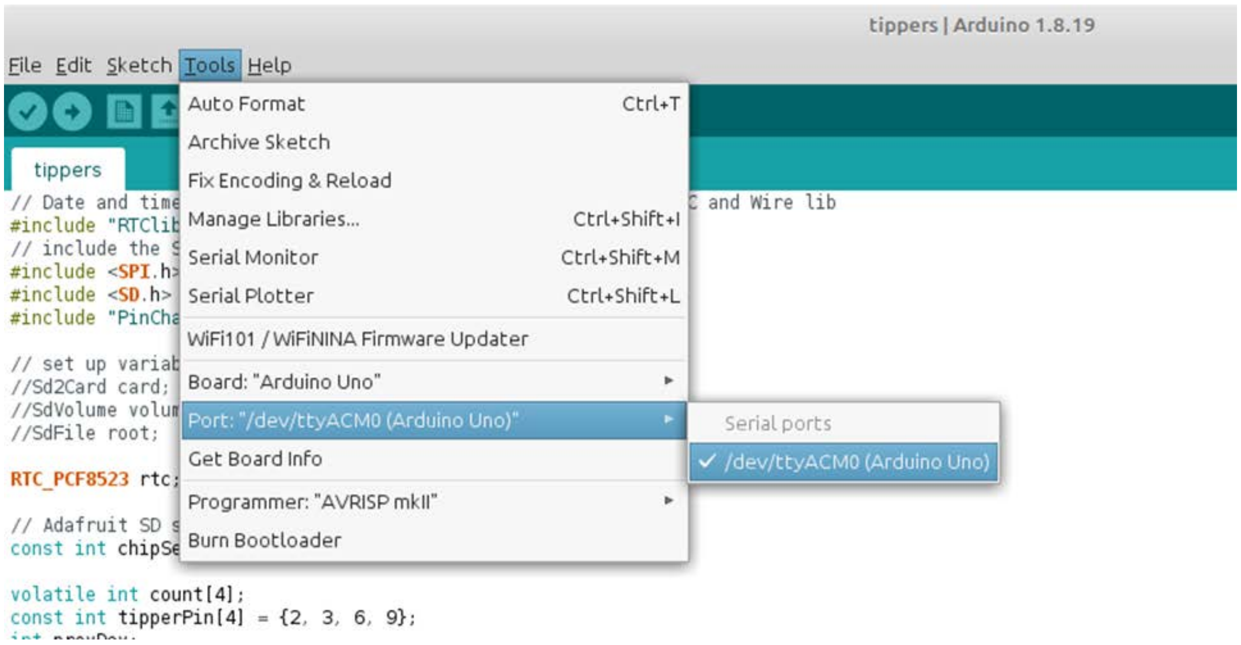
```
Sketch uses 22414 bytes (69%) of program storage space. Maximum is 32256 bytes.
Global variables use 1471 bytes (71%) of dynamic memory, leaving 577 bytes for local variables. Maximum is 2048 bytes.
```

b) Verify the program by clicking on the checkmark button in the top left of the Arduino IDE. You should see a message "Compiling sketch" and then it should change to "Done compiling" at the bottom of the green bar. In the black you should see informational messages and no errors if it compiled properly. If properly compiled, move on to installing.

6. Install Arduino program

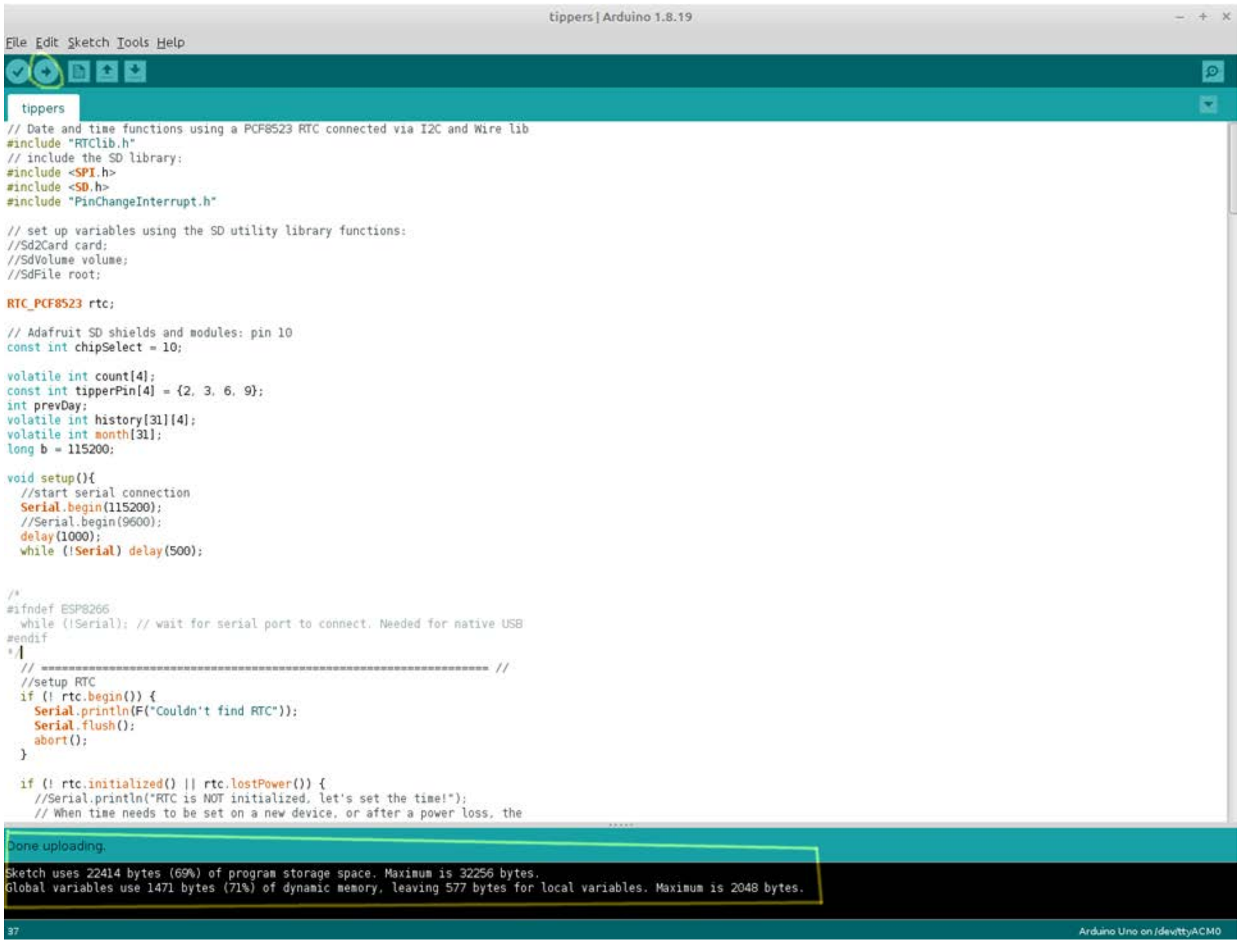
a) Plug the Arduino into the computer via micro USB cable. You should now be able to select Tools – Port from the menubar and see a serial port listed. *Note that serial port names vary between Windows, Linux, and Mac, but select whatever is listed there – it should be the only thing in the menu.*

b) Under Tools – Board, Arduino Uno should be selected. This is the default so it should not require any action but it is good to verify that the proper board is selected.


tippers | Arduino 1.8.19
File Edit Sketch Tools Help
tippers
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "/dev/ttyACM0 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ /dev/ttyACM0 (Arduino Uno)

```
// Date and time                                    C and Wire lib
#include "RTClib
// include the S
#include <SPI.h>
#include <SD.h>
#include "PinCha

// set up variab
//Sd2Card card;
//SdVolume volum
//SdFile root;

RTC_PCF8523 rtc;

// Adafruit SD s
const int chipSe

volatile int count[4];
const int tipperPin[4] = {2, 3, 6, 9};
```

c) Install the program by clicking the right arrow next to the check mark. You should see the bottom label in the green bar say “Uploading”.  When it is done after a few seconds it should say “Done uploading.” and the black area should show informational messages with no errors. If this is correct, your program is now installed!

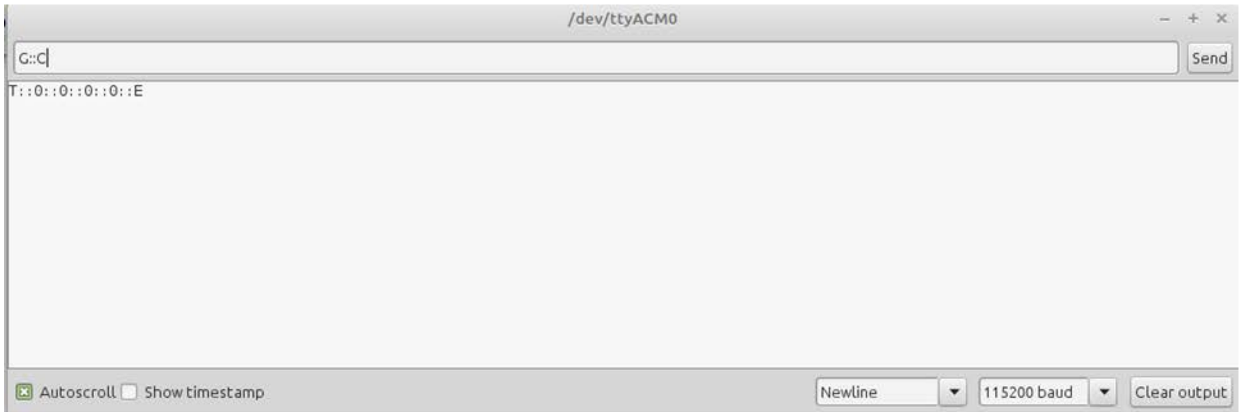


```
// Date and time functions using a PCF8523 RTC connected via I2C and Wire lib
#include "RTClib.h"
// include the SD library:
#include <SPI.h>
#include <SD.h>
#include "PinChangeInterrupt.h"

// set up variables using the SD utility library functions:
//Sd2Card card;
//SdVolume volume;
//SdFile root;

RTC_PCF8523 rtc;

// Adafruit SD shields and modules: pin 10
const int chipSelect = 10;

volatile int count[4];
const int tipperPin[4] = {2, 3, 6, 9};
int prevDay;
volatile int history[31][4];
volatile int month[31];
long b = 115200;

void setup(){
  //start serial connection
  Serial.begin(115200);
  //Serial.begin(9600);
  delay(1000);
  while (!Serial) delay(500);


/*
#ifndef ESP8266
  while (!Serial); // wait for serial port to connect. Needed for native USB
#endif
*/
  // ==================================================================== //
  //setup RTC
  if (! rtc.begin()) {
    Serial.println(F("Couldn't find RTC"));
    Serial.flush();
    abort();
  }

  if (! rtc.initialized() || rtc.lostPower()) {
    //Serial.println("RTC is NOT initialized, let's set the time!");
    // When time needs to be set on a new device, or after a power loss, the
```

```
Done uploading.
Sketch uses 22414 bytes (69%) of program storage space. Maximum is 32256 bytes.
Global variables use 1471 bytes (71%) of dynamic memory, leaving 577 bytes for local variables. Maximum is 2048 bytes.
```

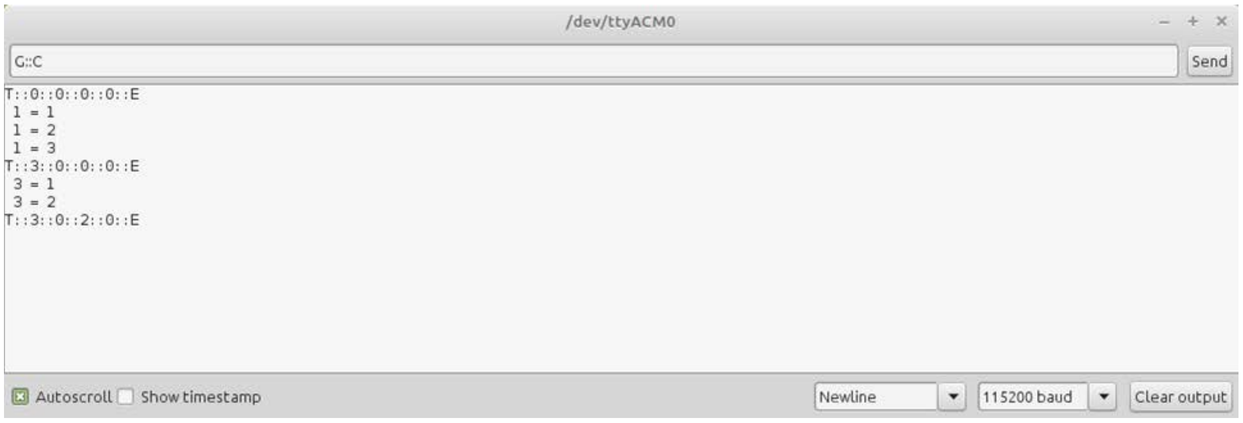
7. Test the output

a) If you select Tools – Serial Monitor to bring up the Serial Window, you can verify that the program is installed. When the Serial Monitor comes up, change the baud rate pulldown in the bottom right to be 115200 baud. Then type G::C into the text field at the top and click Send. You should see a reply T::0::0::0::0::E. If you do, the program is working. You may then try tipping a tipper and you should see the counts displayed (tipper number = total tips for that unit). If you enter G::C again you should see the zeros replaced by whatever the tipper values are for tippers 1, 2, 3, and 4 (hooked up to pins 2, 3, 6, 9). You may now close the Arduino IDE and open the **WaterTips** app on your Android phone and it should connect to the Arduino.



```
G::C
```

```
T::0::0::0::0::E
```



```
G::C
```

```
T::0::0::0::0::E
 1 = 1
 1 = 2
 1 = 3
T::3::0::0::0::E
 3 = 1
 3 = 2
T::3::0::2::0::E
```

## Installing WaterTips app

Follow these steps to download the WaterTips app and install on any Android phone.

1. Open Chrome and go to mybmp.org/app-debug.apk - it should download (or ask if you want to download).

2. Open your Files app and browse to Downloads. There you will see app-debug.apk.zip - for some reason a .zip is added to it.

3. If you have a previous version of app-debug.apk, delete it before the next step

4. Select the ... next to app-debug.apk.zip and rename file to app-debug.apk by removing .zip

5. Click on app-debug.apk to install. If this is the first time you are doing this from a device, it will prompt you that you need to change settings to be able to install this way. Click on Settings on the popup and change the slider to ON to allow Files to install. Then click back and it will prompt you to install. If this is not the first time it will simply prompt you to install. Click Install.

6. The WaterTips icon should appear on your device.

Additionally, if this is the **first time installing** the app on a device, there is a permissions restriction that prevents scanning. To fix that:

1. Open WaterTips

2. Close the app (or just navigate away) and go into Android's Settings (not in the app, but for the phone).

3. Go to Apps & Notifications - you should see WaterTips under recently opened apps - click it (if you don't click See All Apps and scroll down to it and click it).

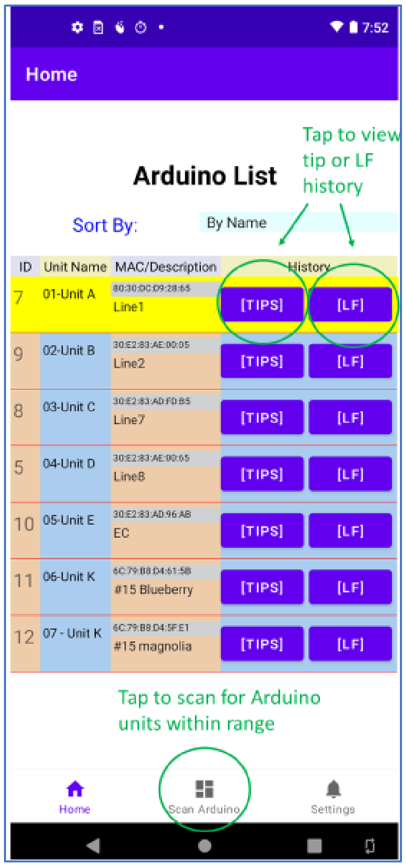
4. Click Permissions - it should say Location under Denied.  Click it and select to allow always.

5. Re-open WaterTips and you should be able to scan Arduinos now.

## How does the app work?

### Arduino scan and initial Arduino setup

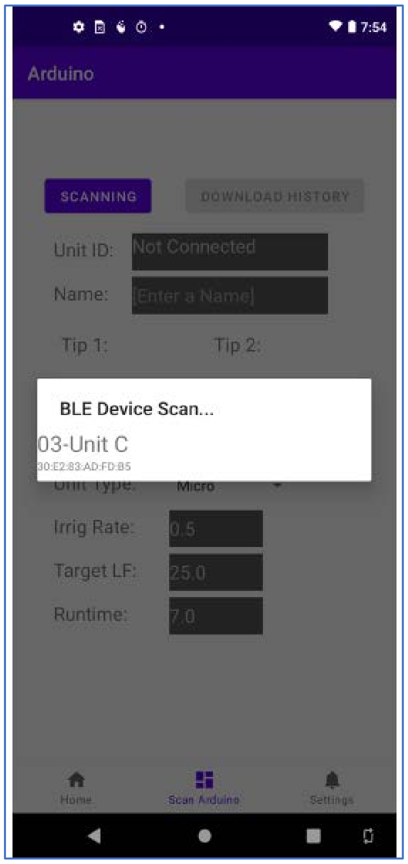
Each Arduino is identifiable by its unique MAC address. When establishing a Bluetooth connection to an Arduino for the first time, the user needs to provide a name (typically an irrigation zone designation) that will be more useful to the user than a MAC address. To connect to an Arduino, tap the 'Scan Arduino' icon and then select the Arduino from the list of available Arduino units within Bluetooth range. Once selected, type in a name (e.g., X1), a description (e.g., #15 podo), irrigation type (micro or sprinkler) and an irrigation rate [inch/hour (sprinkler) or gallon/hour (micro). Note that the user can edit these input values later. After inputting the above information, the user needs to hit 'Add Unit'. Subsequent Bluetooth scans will then show the new name for that Arduino unit.


7:52
Home
Tap to view tip or LF history
Tap to scan for Arduino units within range
Home
Scan Arduino
Settings

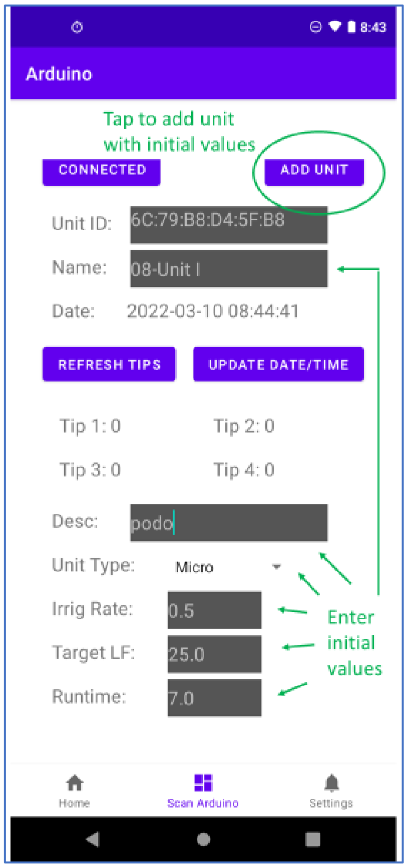
Arduino List

Sort By: By Name

| ID | Unit Name | MAC/Description | History | |
|---|---|---|---|---|
| 7 | 01-Unit A | 80:30:DC:09:28:65<br>Line1 | [TIPS] | [LF] |
| 9 | 02-Unit B | 30:E2:83:AE:00:05<br>Line2 | [TIPS] | [LF] |
| 8 | 03-Unit C | 30:E2:83:AD:FD:B5<br>Line7 | [TIPS] | [LF] |
| 5 | 04-Unit D | 30:E2:83:AE:00:65<br>Line8 | [TIPS] | [LF] |
| 10 | 05-Unit E | 30:E2:83:AD:96:AB<br>EC | [TIPS] | [LF] |
| 11 | 06-Unit K | 6C:79:B8:D4:61:5B<br>#15 Blueberry | [TIPS] | [LF] |
| 12 | 07 - Unit K | 6C:79:B8:D4:5F:E1<br>#15 magnolia | [TIPS] | [LF] |



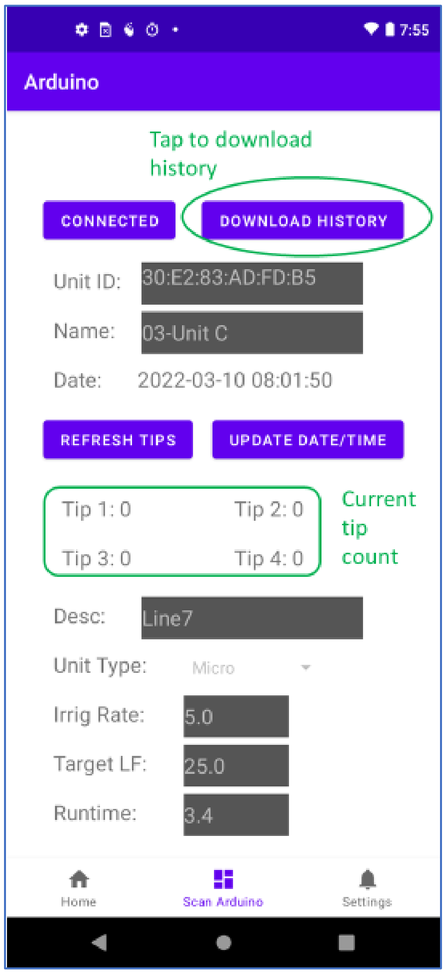
Unit ID: Not Connected
Name: [Enter a Name]
Tip 1: Tip 2:
BLE Device Scan...
03-Unit C
30:E2:83:AD:FD:B5
Unit Type: Micro
Irrig Rate: 0.5
Target LF: 25.0
Runtime: 7.0



Unit ID: 6C:79:B8:D4:5F:B8
Name: 08-Unit I
Date: 2022-03-10 08:44:41
Tip 1: 0 Tip 2: 0
Tip 3: 0 Tip 4: 0
Desc: podo
Unit Type: Micro
Irrig Rate: 0.5
Target LF: 25.0
Runtime: 7.0

### Download history

To acquire tip data, open WaterTips and press 'Scan Arduino' at bottom of screen. Press 'Scan' to display Arduinos within Bluetooth range. Select the desired unit and a Bluetooth connection will be established displaying the current tip counts (counts are reset to 0 at the end of each day) for each of the connected rain gauges. Once connected to the Arduino unit, the 'Scan' box will read 'Connected'. To download tip data, hit 'Download History'. Once download is completed, press 'Connected' to return to 'Scan' function. The download will include daily tip data for all days after the last download.



Unit ID: 30:E2:83:AD:FD:B5

Name: 03-Unit C

Date: 2022-03-10 08:01:50

Tip 1: 0 Tip 2: 0

Tip 3: 0 Tip 4: 0

Desc: Line7

Unit Type: Micro

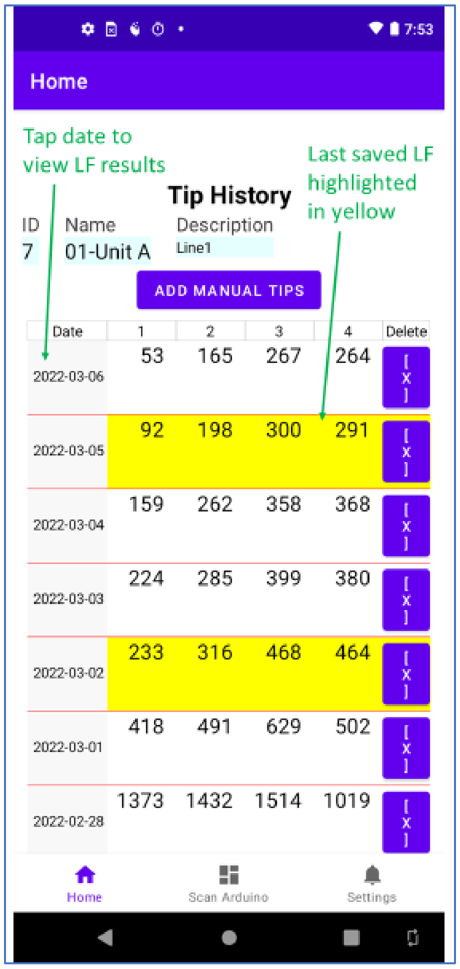
Irrig Rate: 5.0

Target LF: 25.0

Runtime: 3.4

## <u>View tip history and edit and save LF tests</u>

Press 'Home' to view Arduino list. Find the desired unit and press '[tips]' to view tip history (below). The last saved LF test date will be highlighted in yellow. The user would typically want to view tip data for dates after the last saved LF and determine if a new LF test date should be saved and irrigation adjusted. Besides tip data, a decision to save an LF test would consider weather conditions for test days. Once a LF test date is selected for saving, press the date (left column) and a 'Unit Calcs' page is displayed. The 'Unit Calcs' page will display tip data, calculated LF, target LF, current run time (RT) and new run time (RTT). If suspected to be bad, the user can delete a tip value by unchecking a box next to the gauge. The default run time (RT) is assumed to be the RTT (run time adjusted to achieve the target LF) from the previously saved LF test. If different, press 'Override' in "Select RT:" menu and enter an override RT value. Then hit 'Save LF' to save the LF test with the override RT into LF history.



7:53

Home

Tap date to view LF results

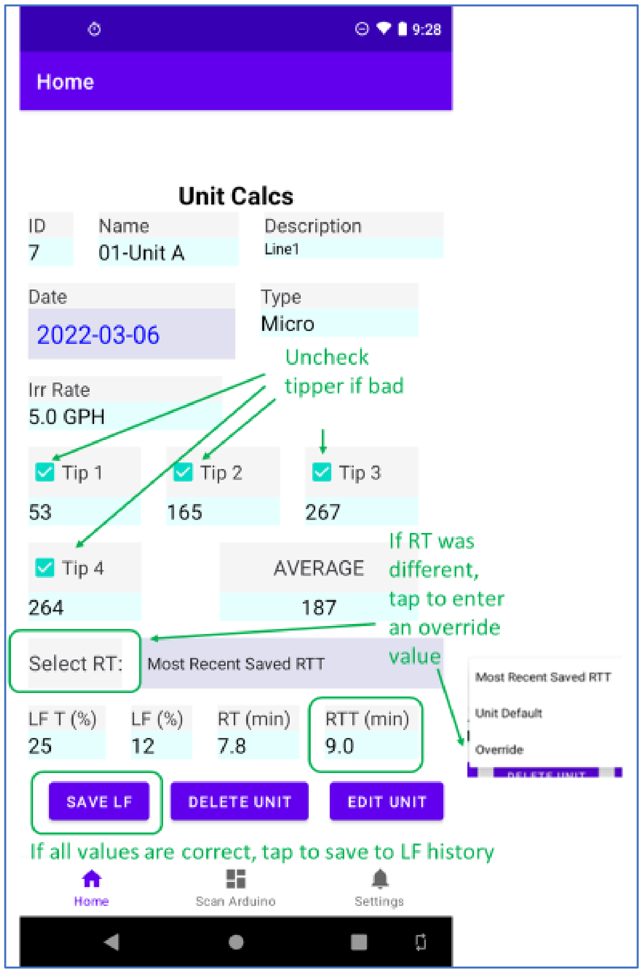
Last saved LF highlighted in yellow

**Tip History**

| ID | Name | Description |
|---|---|---|
| 7 | 01-Unit A | Line1 |

ADD MANUAL TIPS

| Date | 1 | 2 | 3 | 4 | Delete |
|---|---|---|---|---|---|
| 2022-03-06 | 53 | 165 | 267 | 264 | [ X ] |
| 2022-03-05 | 92 | 198 | 300 | 291 | [ X ] |
| 2022-03-04 | 159 | 262 | 358 | 368 | [ X ] |
| 2022-03-03 | 224 | 285 | 399 | 380 | [ X ] |
| 2022-03-02 | 233 | 316 | 468 | 464 | [ X ] |
| 2022-03-01 | 418 | 491 | 629 | 502 | [ X ] |
| 2022-02-28 | 1373 | 1432 | 1514 | 1019 | [ X ] |



9:28

Home

**Unit Calcs**

| ID | Name | Description |
|---|---|---|
| 7 | 01-Unit A | Line1 |

| Date | Type |
|---|---|
| 2022-03-06 | Micro |

Uncheck tipper if bad

Irr Rate
5.0 GPH

| ✓ Tip 1 | ✓ Tip 2 | ✓ Tip 3 |
|---|---|---|
| 53 | 165 | 267 |

| ✓ Tip 4 | AVERAGE |
|---|---|
| 264 | 187 |

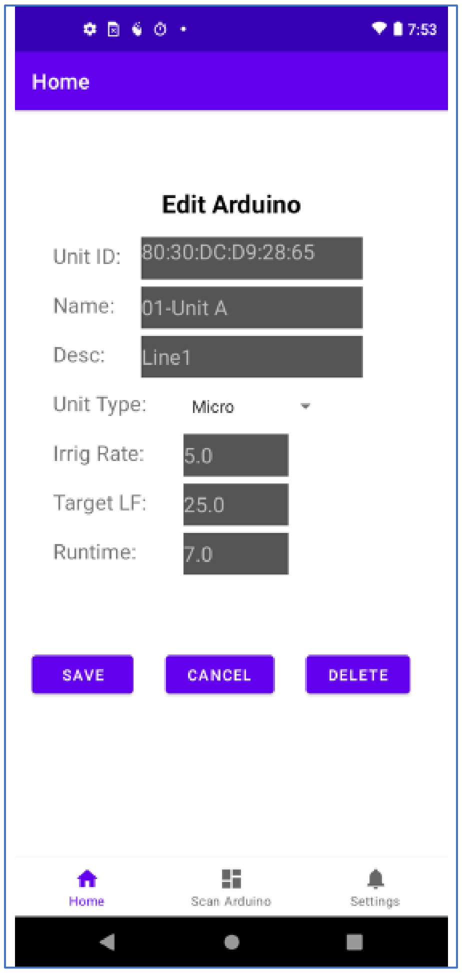
If RT was different, tap to enter an override value

Select RT: Most Recent Saved RTT

| LF T (%) | LF (%) | RT (min) | RTT (min) |
|---|---|---|---|
| 25 | 12 | 7.8 | 9.0 |

If all values are correct, tap to save to LF history

Note that you can also change RT by selecting ‘Edit Unit’. Other inputs can also be changed on the Edit Units page: Unit Name, Description, Irrigation Type, Irrigation rate, target LF (LF T). Press ‘SAVE’ to implement new values and return to ‘Edit Calcs’ page.

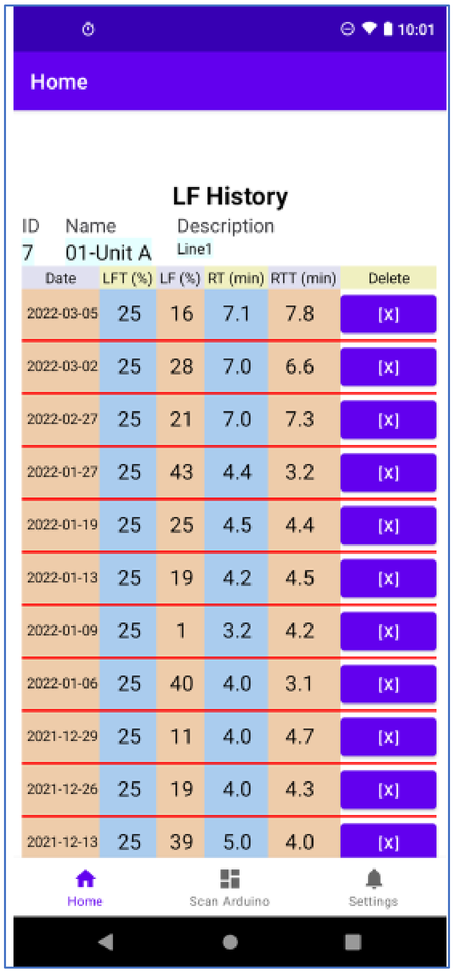


**Edit Arduino**

|  |  |
|---|---|
| Unit ID: | 80:30:DC:D9:28:65 |
| Name: | 01-Unit A |
| Desc: | Line1 |
| Unit Type: | Micro |
| Irrig Rate: | 5.0 |
| Target LF: | 25.0 |
| Runtime: | 7.0 |

**<u>View LF history and adjust irrigation</u>**

Press ‘Home’ to view Arduino list. Find the desired unit and press ‘[LF]’ to view LF history. The most recent LF test will be displayed at the top of the list. The RTT value displayed will be the new run time to implement. If multiple irrigation cycles are used, the RTT value will need to be divided by the number of cycles to calculate the run time for each cycle.



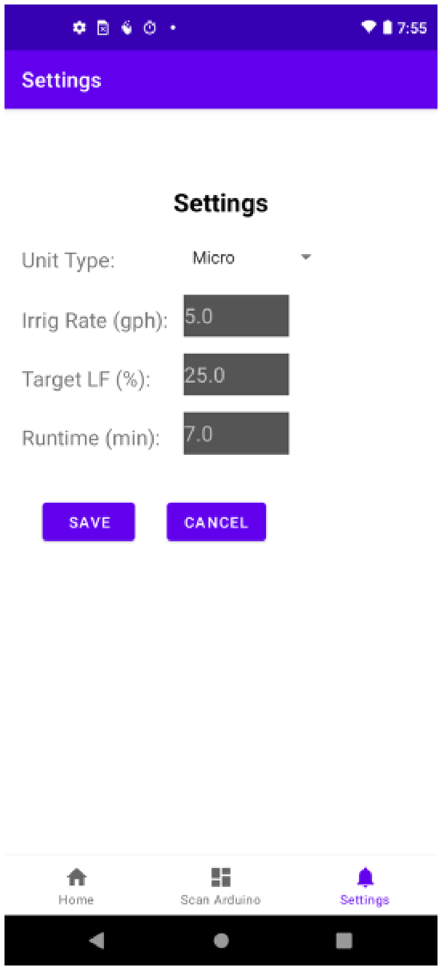
LF History

| ID | Name | Description |
|---|---|---|
| 7 | 01-Unit A | Line1 |

| Date | LFT (%) | LF (%) | RT (min) | RTT (min) | Delete |
|---|---|---|---|---|---|
| 2022-03-05 | 25 | 16 | 7.1 | 7.8 | [x] |
| 2022-03-02 | 25 | 28 | 7.0 | 6.6 | [x] |
| 2022-02-27 | 25 | 21 | 7.0 | 7.3 | [x] |
| 2022-01-27 | 25 | 43 | 4.4 | 3.2 | [x] |
| 2022-01-19 | 25 | 25 | 4.5 | 4.4 | [x] |
| 2022-01-13 | 25 | 19 | 4.2 | 4.5 | [x] |
| 2022-01-09 | 25 | 1 | 3.2 | 4.2 | [x] |
| 2022-01-06 | 25 | 40 | 4.0 | 3.1 | [x] |
| 2021-12-29 | 25 | 11 | 4.0 | 4.7 | [x] |
| 2021-12-26 | 25 | 19 | 4.0 | 4.3 | [x] |
| 2021-12-13 | 25 | 39 | 5.0 | 4.0 | [x] |

## Settings

The setting page includes defaults that will be used when initially adding an Arduino. These can be changed when initially adding the Arduino or later using the 'Edit Calcs' page in Tip History.



Settings

Unit Type: Micro

Irrig Rate (gph): 5.0

Target LF (%): 25.0

Runtime (min): 7.0

## Other app features

### Clock accuracy

A 3V clock battery on the data logging shield maintains date and time of day. It can be updated with the app by connecting to the Arduino and selecting ‘UPDATE DATE/TIME’. The Arduino will then set the clock to the date and time of the mobile device.

### Checking tippers

When troubleshooting tippers, a refresh feature allows you to update tip counts without having to rescan the Arduino. After connecting to the Arduino and manually tilting the suspected tipper to cause tips, confirm that tips are being recorded by tapping ‘REFRESH TIPS’ and look for an increase in tip counts. If no tips are being recorded, then either the tipper needs maintenance or the connection to the Arduino needs to be checked.

### Entering manual tip data

There is a provision to manually enter tip data if another system was used to monitor tips. In the 'Tip History' page, tap 'ADD MANUAL TIPS'. Enter the prompted date and tip values and tap 'ADD'.

**<u>Downloading tip data from SD cards</u>**

If power is removed from Arduino, data in the Arduino's memory will be lost. However, the data (date, tip1, tip2, tip3, tip4) can be recovered from the SD card using the following steps:

1) Paste the TipsTo CSV.class file into a folder that has java.exe in its PATH. At UF, place the .class file in the bmpinstall/bin folder (this folder is used for java agents associated with CIRRIG)

2) Open DOS command prompt and change directory to the folder where the .class file is located: e.g. cd C:\bmpinstall/bin

3) Insert SD card in USB slot and determine the directory used

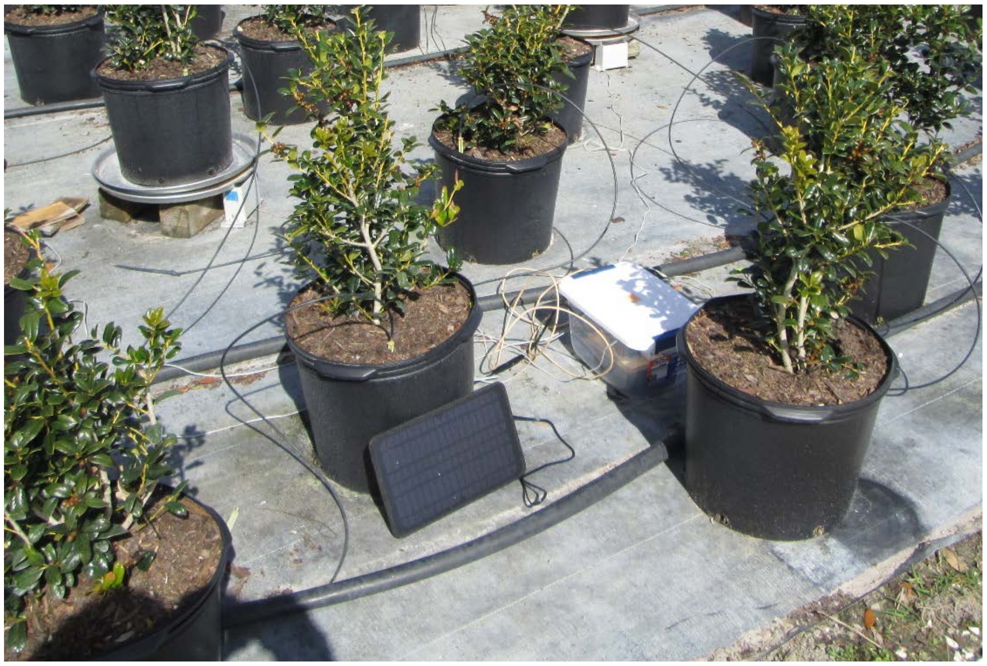
4) enter the following command replacing "G:" with the directory used by the SD card
```
javaw.exe TipsToCSV G:/ output.csv
```

5) look for the .csv output file in the same directory where the class file is located

## Maintenance

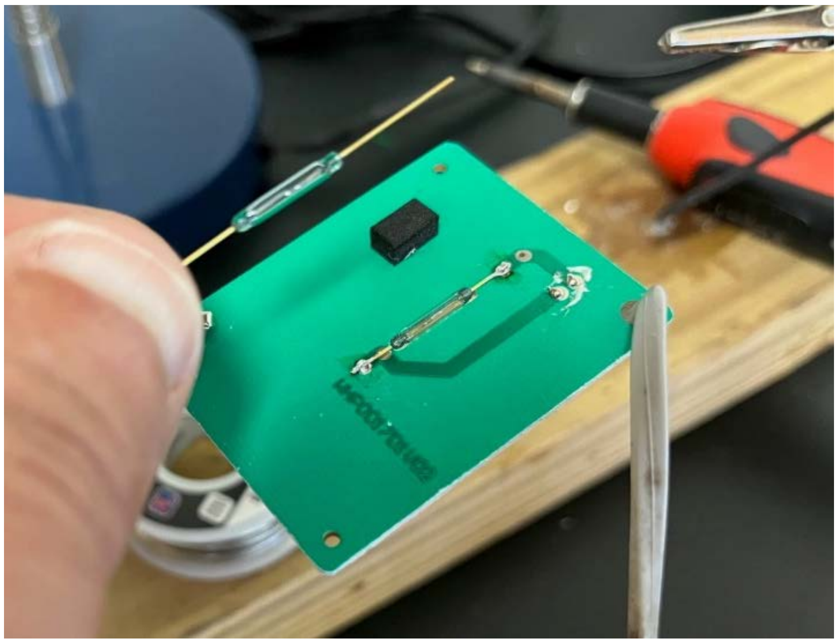
**<u>Battery life and solar charging</u>**

A 12V sealed-lead acid battery rated at 7-9 amp-hours will provide enough power for only six days. We found that a 5-watt, 12V solar charger connected in parallel will maintain battery power. A 10-watt solar charger may be needed in less sunny locations or in the short days of winter.



**Replacing reed switches**

The reed switch in the tipper can become fused in a closed position rendering it functionless. To fix this problem, a new replacement switch can be easily soldered on to the circuit board inside the tipper. Remove the defective switch by heating up the solder joints and pulling switch off the board.  Cut the replacement switch leads to match the lengths of the defective one and solder the new one back on to circuit board.



## Hardware costs*

Cost for four tippers in sprinkler zone = $153 ($128 Arduino +$25 collectors)

Cost for four tippers in micro zone = $197 ($128 Arduino +$69 collectors)

| Phone | | | | | | |
|---|---|---|---|---|---|---|
| **Component** | **Units** | **$/no. unit** | **$/unit** | **company/description** | **source** | **total $** |
| Android phone | 1 | 179.99/1 | 179.99 | Moto G Power 4/64GB \| 48MP Camera \| Gray | amazon.com | 179.99 |
| Case | 1 | 24.9/1 | 24.90 | Foluu flip magnetic case | dfrobot.com | 24.90 |
| | | | | | | **204.89** |
| **Arduino** | | | | | | |
| **Component** | **Units** | **$/no. unit** | **$/unit** | **company/description** | **source** | **total $** |
| Rain gauge | 4 | 15/1 | 9.35 | MISOL (price for 50 units) | misolie.net | 37.40 |
| Microprocessor | 1 | 24.9/1 | 24.90 | DFRobot Bluno - Arduino compatible (product #1044) | dfrobot.com | 24.90 |
| Data-logging shield | 1 | 13.95/1 | 13.95 | Data logging shield for Arduino (product #1141) | adafruit.com | 13.95 |
| Shield stacking headers | 1 | 1.6/1 | 1.60 | Shield stacking headers R3 compatible (product #85) | adafruit.com | 1.60 |
| Barrel connector | 1 | 0.95/1 | 0.95 | 5.5/2.1 mm Barrel connector plug | adafruit.com | 0.95 |
| 12V Sealed Lead Acid Battery | 1 | 84.14/4 | 21.03 | ML9-12 - 12 Volt 9 AH SLA Battery | amazon.com | 21.03 |
| 5W solar charger | 1 | 16.93/1 | 16.93 | Renogy 5W 12V Portable Solar Panel Battery Maintainer Trickle Charger | amazon.com | 16.93 |
| Micro SD cards | 1 | 8.48/2 | 4.24 | Gigastone 8GB Micro SD Card, 80MB/s Micro SDHC Class 10 | amazon.com | 4.24 |
| Battery for shield | 3 | 5.79/10 | 0.58 | SURPOWER CR1220 3v Lithium Battery | amazon.com | 1.74 |
| Jumper wires | 8 | 6.96/120 | 0.06 | ELEGOO 120pcs Dupont Wire Jumper Wires | amazon.com | 0.48 |
| Misc wires/connectors/tote | 1 | 5/1 | 5.00 | | | 5.00 |
| | | | | | | **128.22** |
| **Leachate collectors for 11"-diam. Trade #3 container - sprinkler** | | | | | | |
| **Component** | **Units** | **$/no. unit** | **$/unit** | **company/description** | **source** | **total $** |
| Aluminum pizza pan | 4 | 4.21/pan | 4.21 | American Metalcraft A4011 11" x 1" Std Wt aluminum straight-Sided | webrestaurant.com | 16.84 |
| Wood support | 8 | 6.68/8 ft | 0.75 | 1" x 6" lumber | Lowes | 6.00 |
| Stainless screws | 24 | 5.98/75 | 0.08 | 1 5/8" stainless screws | lowes.com | 1.92 |
| | | | | | | |
| **2-Pan leachate collectors for 17"-diam. trade #15 containers - microirrigation** | | | | | | |
| **Component** | **Units** | **$** | **$/unit** | **company/description** | **source** | **total $** |
| Aluminum pizza pans ($5-10) | 8 | 5.99/1 | 5.99 | American Metalcraft ADEP17 17" x 1" Std Wt Alum Tapered/Nesting | webrestaurant.com | 47.92 |
| Hose filter | 4 | 9.98/80 | 0.12 | Bcoress 80 pcs stainless steel filter hose washers inlet for 3/4 inch hose | lowes.com | 0.50 |
| Filter sand | 0.8 | 12.99/50# | 0.26 | HTH pool filter sand | acehardware.com | 0.21 |
| Wood support | 24 | 6.68/8 ft | 0.84 | 1 x 6 inch pressure treated lumber | lowes.com | 20.04 |
| Stainless screws | 24 | 5.98/75 | 0.08 | 1 5/8" stainless screws | lowes.com | 1.92 |
| | | | | | | **68.67** |

| **Summary** | **$** |
|---|---|
| Phone | 205 |
| Arduino | 128 |
| Leachate collectors - sprinkler zone | 25 |
| Leachate collectors - micro zone | 69 |

*3-11-22

## Disclaimers