# Using the Adafruit Stemma Soil Sensor

With Arduino Uno, UART, and C

**Hardware Required**.....1
**Prerequisite Knowledge**.....1
**Rational of Approach**.....2
**Circuit Design**.....2
**Arduino Setup**.....3
**Beaglebone UART Initialization**.....3
**Beaglebone Functions**.....4
**Troubleshooting**.....4
**Appendix**.....4
Arduino Sketch.....4
BeagleBone Initialization Function.....5
Communication Helper Functions.....7
Read Temperature Function.....8
Read Moisture Function.....8
**References**.....8

## Hardware Required

1. Arduino Uno [1]
2. Beaglebone
3. Adafruit I2c Stemma Soil Sensor [2] with its Seesaw Library [3]
4. Level Shifter

## Prerequisite Knowledge

The user should be comfortable with cross compiling to Beaglebone and basic circuit design. The user should also be familiar with how an Arduino works. If not a brief high level description:

The Arduino is a microcontroller that operates in real time. It operates using sketches. The basic Arduino template consists of a Setup() and a loop() function. Setup is where one would establish which pins are used as input or output on startup. Serial and I2C would also be setup during this time. This part of the code runs only when the Arduino is powered up in the beginning.

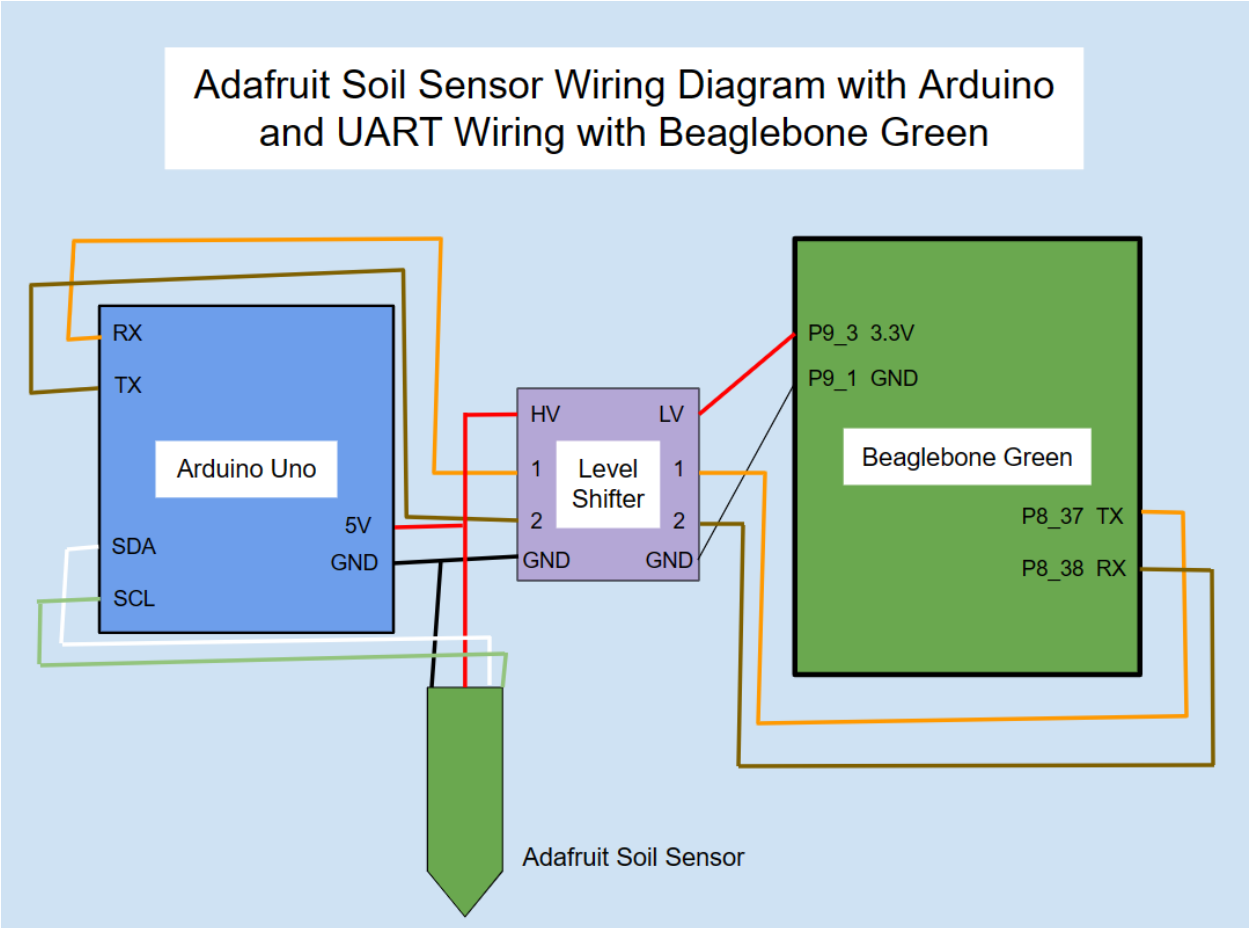
The loop function is self-explanatory. As long as there is power to the board, it will loop this code in an indefinite while(true) loop.

## Rational of Approach

The reason for the approach of relying on two boards is because Beaglebone is unable to read from address 0x36 and get any information from the Adafruit Stemma Soil Sensor. When trying to read the information using i2cdump they would all come out as max values. It reportedly worked with Python but the library provided by Adafruit was unable to execute on the board at the time of this project. However, on Arduino it ran very simply with the Adafruit_seesaw library.

As the device worked flawlessly with Arduino and it is very simple to get readings from it, in order to incorporate Beaglebone, a method of communication needed to be established. UART Serial communication proved to be very effective in getting the two devices to talk and share the readings between the two boards.

## Circuit Design



The Adafruit Soil Sensor already comes with colour coded wiring on their device as well as their product page outlining which cable belongs where. It is a great reference for getting you up and running with an Arduino Uno, complete with an example sketch for readings.

The two devices are connected via a level shifter as an Arduino will be operating on 5V while a Beaglebone's GPIO pins operate at 3.3V. This means the voltage needs to be stepped up or down depending on the direction of transfer. The soil sensor operates the best when it is powered by 5V by the Arduino as opposed to 3.3V.

Also take special note of the TX and RX pins on the respective boards. TX means transfer and by following the orange line, you can see that the TX on the Beaglebone connects with the RX, or receive on the Arduino. The other direction is also true with the TX on the Arduino connected with the RX on the Beaglebone. This wiring of directions is really important for effective communication.

Another reminder that when uploading sketches to the Arduino using the Arduino IDE, you will need to momentarily disconnect the RX and TX jumpers from the Arduino until the transfer is complete. Use colour coded jumpers to remember how to connec them back when starting the application on Beaglebone.

## Arduino Setup

Upload the Arduino Sketch in the appendix. Note that in order to upload this sketch to the board, you will need to install Arduino IDE and the drivers to connect with Arduino IDE. This can be done either with Windows or Linux. In order for the sketch to upload to the Arduino, you will need to momentarily disconnect the RX and TX pins (as described above in the block diagram) as they disable communication to the board. Once the sketch has been uploaded successfully, reconnect those pins correctly to ensure communication between the Beaglebone and Arduino.

There are global variables set with arbitrary values, but there are also two variables that are used for calibration for a proper soil moisture percentage reading. These values were taken on a sunny day in Vancouver. The readings were taken in air and in extremely moist sale to establish a minimum and maximum value. If the weather is different, these values could possibly vary and can be changed to get a more accurate reading.

The Arduino sketch will wait in its loop function using a busy wait until the Beaglebone sends a character byte message to receive a measurement from the soil sensor.

## Beaglebone UART Initialization

A module for Beaglebone is attached in the appendix. The key part to getting communication between the Arduino and Beaglbeone is the connection of the respective TX to RX pins. To make this connection the termios.h library is used [4]. Ensure that there is a wait time to allow

for full initialization before communication or there will be errors in communication. The Beaglebone Green keeps its UART access files under /dev/ttyS<1-5>, you can find out the pins that correspond with each UART channel by visiting [5].

## Beaglebone Functions

Also present in the Appendix is the readTemp and readMoisture functions. What is important for proper communication is that there is a wait time between the Beaglebone sending a command and when it can receive a response from the Arduino. The optimal timing discovered through testing was 200 ms. If there are errors with the Beaglebone being unable to read, then increasing these timings may help.

## Troubleshooting

1. I am just getting an unable to read from serial port message/there is no communication!
   Ensure that the pins connecting the Beaglebone to the Arduino are connected correctly (Arduino TX to Beaglebone RX, Arduino RX to Beaglebone TX) and that you are using reasonable timing for the Beaglebone to wait for a response to/from the Arduino, we found that 200ms seemed to work well.

2. My moisture level seems to be way off what it should be!
   Your sensor might require some calibration, take measurements with the sensor in wet soil and when it is dry in the air, that should help you get a sense of how wet the soil is and make a simple linear relation/percentage to send back to the Beaglebone.

## Appendix

### Arduino Sketch

```
#include <Wire.h>
#include "Adafruit_seesaw.h"


Adafruit_seesaw ss;
// Make sure to unplug pins 1 and 0 for communication
// Initial arbitrary values
int temp = 23;
int moisture = 50;

//Calibration variables
float waterMoistureLevel = 800;
float airMoistureLevel = 350;
```

```
char msg = 0;
char response = 0;
int confirm = 1;
int fail = 0;

void setup (void)
{
  Serial.begin(115200);
  ss.begin(0x36);

}



void loop (void)
{

  while (!Serial.available());
  msg = Serial.read();

  if (msg == 't'){
    //Serial.print("temperature: ");
    temp = ss.getTemp();
    Serial.println(temp);
   // Serial.println("*C");
  }else if (msg =='m') {
    //Serial.print("moisture level: ");
    uint16_t sensorRead = ss.touchRead(0);
    moisture = (((float)sensorRead -
airMoistureLevel)/(waterMoistureLevel-airMoistureLevel))*100;
    if (moisture < 0) {
      moisture = 0;
    } else if (moisture > 100) {
      moisture = 100;
    }
    Serial.println(moisture);
  }
}
```

## BeagleBone Initialization Function

```
#include <fcntl.h>
#include <termios.h>
```

```
#define SOILSENSOR_RX_PIN "p8.38"
#define SOILSENSOR_TX_PIN "p8.37"
#define SOILSENSOR_UART "5"
#define TTY_FILE "/dev/ttyS" SOILSENSOR_UART
#define WRITE_WAIT 200

int serial_fd;

void SoilSensor_initialize(void)
{
    // Set the RX and TX pins on BBG to UART
    runCommand(RX_CONFIG_CMD);
    runCommand(TX_CONFIG_CMD);
    //runCommand is a function provided by Dr. Brian

    // Set up the Serial connection for read/write
    serial_fd = open(TTY_FILE, O_RDWR | O_NOCTTY | O_NDELAY);
    if (serial_fd < 0) {
        printf("soil_sensor: Error opening fd for %s.\n", TTY_FILE);
    }

    // Termios flags from:
https://stackoverflow.com/questions/55202385/termios-c-serial-read-no
t-reading-arduino-serial
    struct termios tty; // create termios config
    memset(&tty, 0, sizeof tty);
    if (tcgetattr(serial_fd, &tty) != 0) {
        printf("soil_sensor: Unable to get termios attributes.\n");
    }

    // set flags
    tty.c_cflag &= ~PARENB;
    tty.c_cflag |= CS8; // 8 bits per byte
    tty.c_cflag |= CREAD | CLOCAL;
    tty.c_lflag &= ~ICANON;
    tty.c_lflag &= ~ECHO; // disable echo
    tty.c_lflag &= ~ECHOE; // disable erasure
    tty.c_lflag &= ~ECHONL; // disable new-line echo
    tty.c_lflag &= ~ISIG;
    tty.c_iflag &= ~(IXON | IXOFF | IXANY);
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR |
IGNCR | ICRNL);
    tty.c_oflag &= ~OPOST; // prevent special interpretation of
certain bytes like new-line
```

```
    tty.c_oflag &= ~ONLCR; // prevent conversion of new-line to line
feeds
    tty.c_cc[VTIME] = 50; // wait 5 seconds to read
    tty.c_cc[VMIN] = 1; // set blocking or non-blocking
    cfsetispeed(&tty, B115200);
    cfsetospeed(&tty, B115200);

    // apply flags for termios
    if (tcsetattr(serial_fd, TCSANOW, &tty) != 0) {
        printf("soil_sensor: Failed to apply termios config.\n");
    }

}
```

## Communication Helper Functions

```
// Read a value from the arduino soil sensor
char* SoilSensor_read(int *length)
{
    char *read_buff = malloc(1024);

    int result = read(serial_fd, read_buff, 1023);
    if (result < 0) {
        printf("soil_sensor: Unable to read from serial port.\n");
    }
    read_buff[result] = '\0';
    *length = result;
    return read_buff;
}

// Read a value from the arduino soil sensor
void SoilSensor_write(char *com)
{
    int result = write(serial_fd, com, sizeof(com));
    if (result < 0) {
        printf("soil_sensor: Unable to write from serial port.\n");
    }
    sleepForMs(WRITE_WAIT);
}
```

### Read Temperature Function

```
int SoilSensor_readTemp(void)
{
    char c = 't';
    SoilSensor_write(&c);
    int l = 0;
    char *temp = SoilSensor_read(&l);
    soil_temp = atoi(temp);

    return soil_temp;
}
```

### Read Moisture Function

```
int SoilSensor_readMoisture(void)
{
    char c = 'm';
    SoilSensor_write(&c);
    int l = 0;
    char *moistLevel = SoilSensor_read(&l);
    soil_moisture = atoi(moistLevel);

    return soil_moisture;
}
```

## References


[1] Arduino. “Arduino Uno Rev3 Datasheet.” [Online]. Available: https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf. [Accessed March 2023]

[2] Adafruit. “Adafruit Stemma Soil Sensor - I2C Capacitive Moisture Sensor - JST PH 2mm.” [Online]. Available: https://www.adafruit.com/product/4026. [Accessed March 2023]

[3] Adafruit. “Adafruit_seeaw.” [Online]. Available: https://github.com/adafruit/Adafruit_Seesaw. [Accessed February 2023]

[4] Stackoverflow. “Termios c++ serial read not reading arduino serial.” [Online]. Available: https://stackoverflow.com/questions/55202385/termios-c-serial-read-not-reading-arduino-serial. [Accessed March 2023]

[5] MathWorks. “BeagleBone Black Pin Map.” [Online.] Available: https://www.mathworks.com/help/supportpkg/beagleboneio/ug/beaglebone-black-pin-map.html