adafruit learning system

Reverse Geocache Box

Created by Tyler Cooper

Guide Contents

Overview

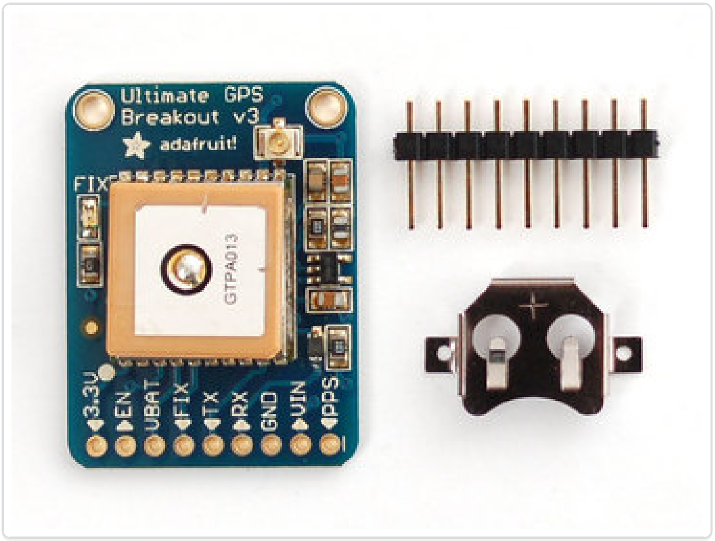
Adafruit customer Kenton Harris used a reverse geocaching box to propose to his girlfriend (she said yes). After the project build and successful proposal, he shared how he built the reverse geocache with us. We thought others may be interested in building a similar box for their own proposals, or for fun!

A reverse geocaching box works similar to other geocaching devices. It will guide you to certain coordinates on earth and instead of finding a geocache located at those coordinates, you carry the box with you, and it opens for you at a predetermined destination. This is also a great way to do a scavenger hunt (making someone go to multiple locations before the box opens).

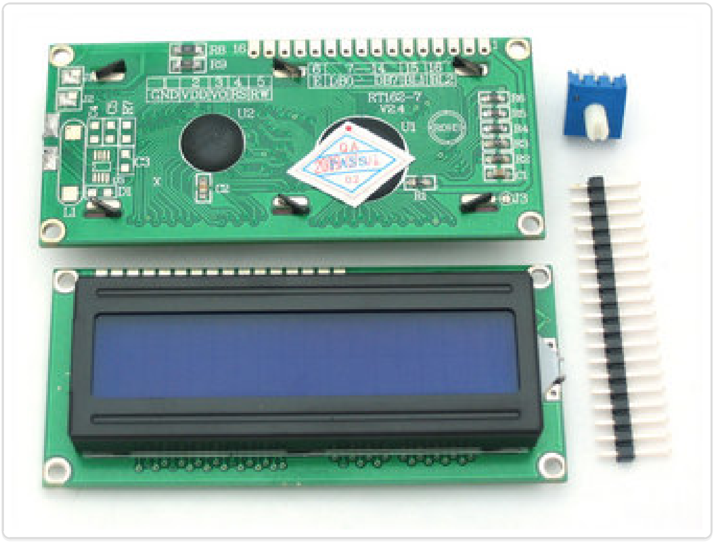
Parts and Tools

To get started with your reverse geocache build, here are the required parts:

An Arduino UNO (http://adafru.it/50) - The brains of the reverse geocache.

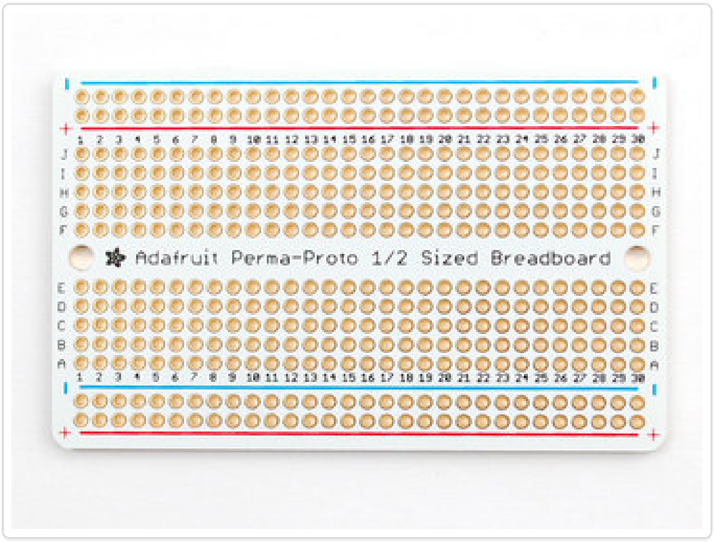


Adafruit Ultimate GPS Module (http://adafru.it/746) - This little guy will tell us where we are, and where we need to go.



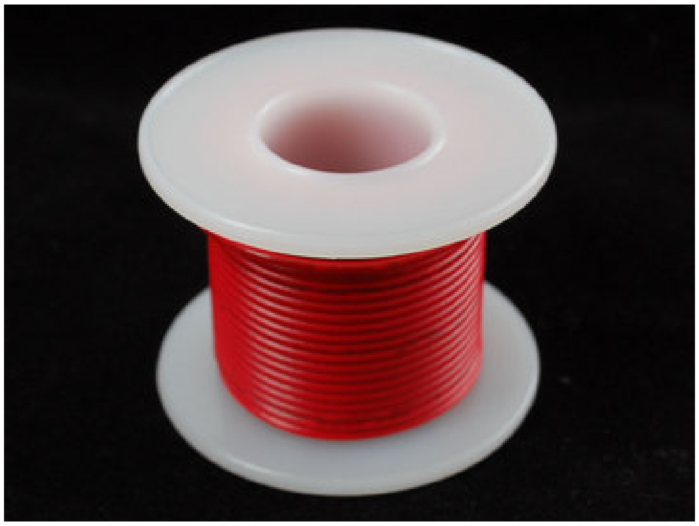
A Standard 16x2 LCD (http://adafru.it/181) - This will be used to communicate with the person using the reverse geocache box. You can have it help people navigate to the destination, give them a clue, or ask them to marry you. :)

A Micro Servo (http://adafru.it/169) - This little servo will be used to lock and unlock the box with the Arduino.



Prototyping Board (http://adafru.it/571) - Assemble and solder your electronics to this proto-board so it doesn't fall apart in the field.

A Power Switch (http://adafru.it/915) - Kenton prefers the Adafruit Waterproof Metal On/Off Switch (with blue LED ring).



Hook-Up Wire (http://adafru.it/288) - For connecting all of the electronics together.

A Battery (http://adafru.it/67) - Power the whole thing with this 9V battery holder.

A Box - Something to enclose the project in. Kenton made his out of cherry, but you can use just about anything. Just don't use a metal box, as the metal will block the GPS signals.

Also, you will need a few tools and supplies:

A Soldering Iron (http://adafru.it/180) - Use this to solder the electronics to the proto-board. If you have never soldered before, check out our soldering tutorial here (http://adafru.it/aTk).

A Spool of Solder (http://adafru.it/145)

Solder Wick (http://adafru.it/149) - To fix your soldering mistakes the easy way.

Wire Cutter & Stripper (http://adafru.it/527)

Electronics Assembly

Understanding the Components

We're going to start by reading the tutorials for each of the components separately, then we are going to combine them to make our reverse geocache box.

- Ultimate GPS Tutorial:http://learn.adafruit.com/adafruit-ultimate-gps (http://adafru.it/aTl)
- Character LCD: http://learn.adafruit.com/character-lcds/ (http://adafru.it/aTm)
- Servo: http://www.arduino.cc/en/Reference/Servo (http://adafru.it/aOD) - It's pretty simple. You just tell it what angle you want the servo to go to, and it obliges.

Read through these links and get to know the basics of how they work. Use a breadboard and examples to experiment and get familiar with them.

No really, go read them. It's important.

Arduino Pins

Now that you've read how the LCD, GPS, and servo work, let's combine them. The serial Library uses digital pins 0 and 1, so we want to stay away from those. This allows us to debug the Ardunio with Serial.print() and see what's going on in the program. The GPS breakout uses digital pins 2 and 3, just like in the Ultimate GPS tutorial. The LCD uses 6 pins, and the Servo motor uses 1 pin.

I put the Servo on Pin 9, and I put the LCD in pins 7,8,6,10,11,and 12 (I had to move the third LCD pin from pin 9 to pin 6 to accommodate the servo).

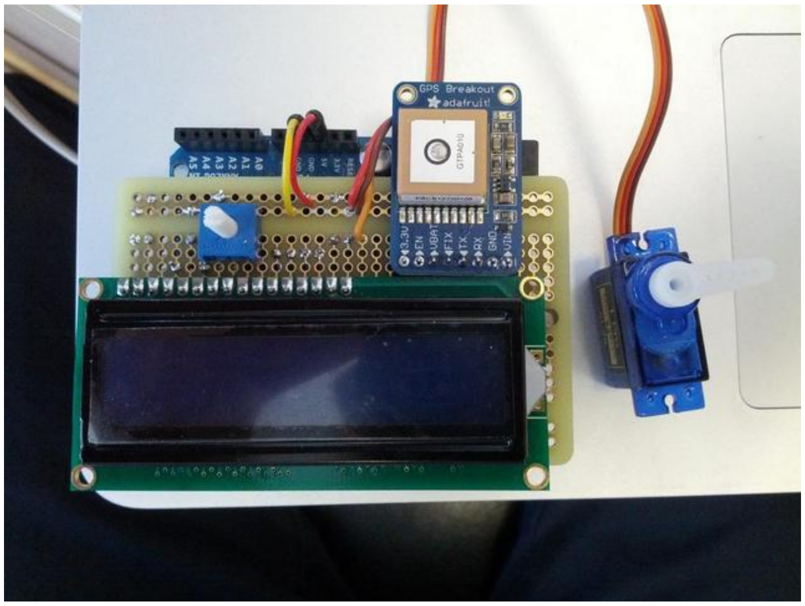
NOTE: The Servo library and the SoftwareSerial library (which the GPS uses to spit out data) do NOT play nice with each other. To fix this, we're going to install an older version of the servo library that doesn't interfere with SofwareSerial.

That library is located here: http://arduiniana.org/libraries/pwmservo/ (http://adafru.it/aTn)

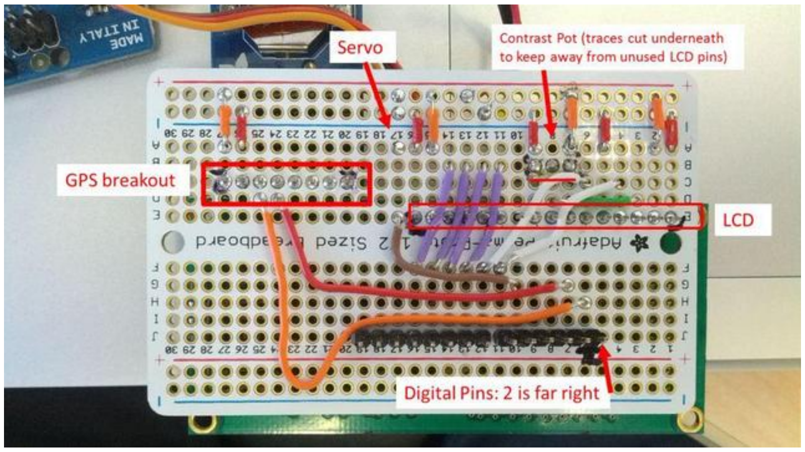
Download version 2 (by Paul Stoffregen) and install it into the libraries folder in your Arduino folder, just like you did with the Adafruit_GPS folder in the Ultimate GPS tutorial. This library is called PWMServo. It requires the servo to be on Pin 9 or 10 (which is why you see the change above). Otherwise, you interface with it similarly, you just have to use servoLatch.attach(SERVO_PIN_A); when you attach the servo pin.

Wire It Up

Now, lay out all of your components. I tried to make the total size as small as possible, and fit into the 5" x 5" box I made. You can see my layout below. The Uno is on the bottom, with two wires coming off to power my proto board. The LCD does not use pins 7, 8, 9, or 10, so I chose to put the contrast potentiometer there. I cut the traces to those pins so I didn't advertently send a signal to the LCD.



Here’s the underside of the proto board. You can see the row of male headers at the bottom. These plug into the digital pins in the Arduino (note: The servo is connected to pin 6 here. It was later swapped to pin 9).

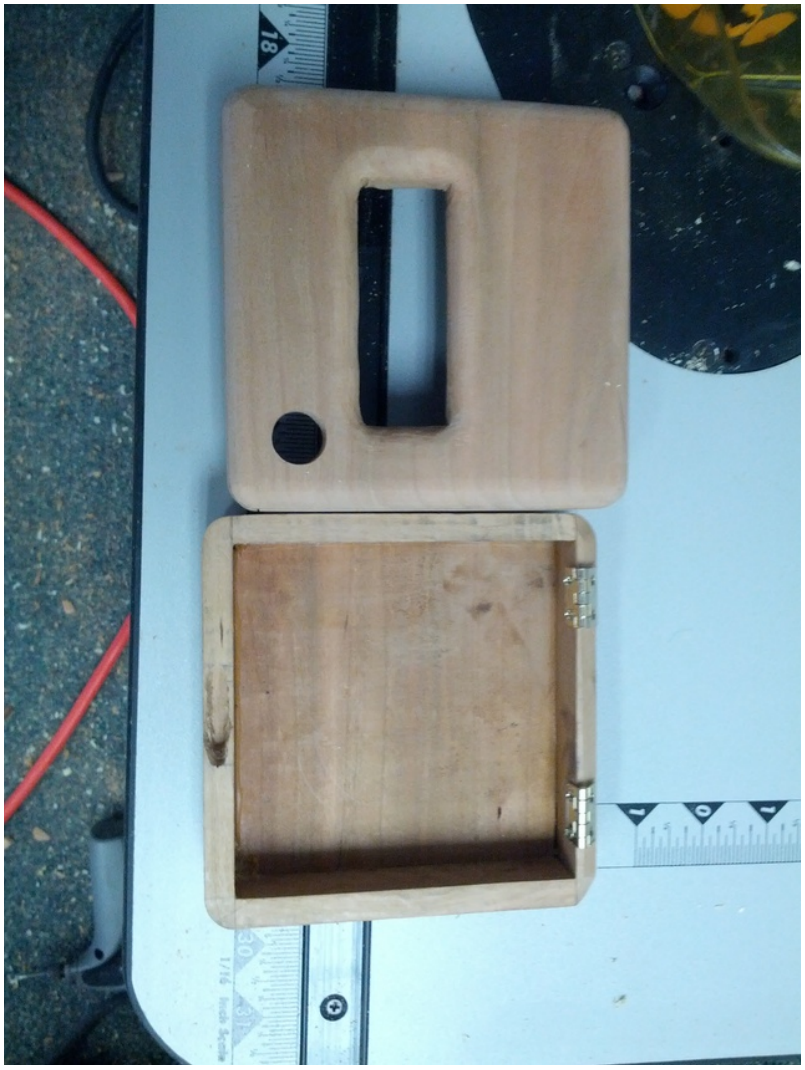

Servo
Contrast Pot (traces cut underneath to keep away from unused LCD pins)
GPS breakout
LCD
Digital Pins: 2 is far right

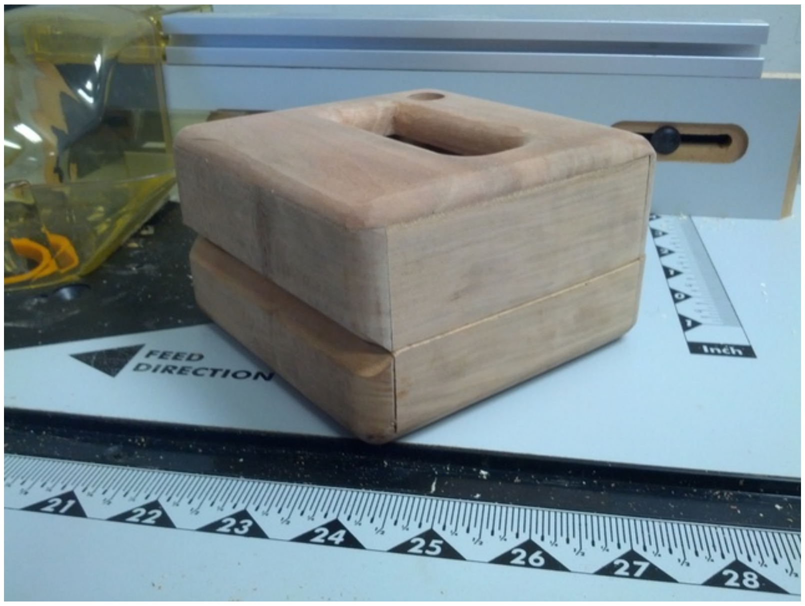
Now you’re ready to melt some solder and connect you electronics! Solder the LCD, GPS, and servo pins into the protoboard. Then connect jumpers to your digital pin headers. Solder the contrast pot into the slots not used by the LCD (I used 7, 8, and 9) and cut the trace to the LCD pin. You can also just not attach headers to the LCD for pins 7-10. I installed a jumper from the pot wiper to pin 3 of the LCD (green wire).

NOTE: If you’re using the perma-proto board, you’ll notice that the digital pins on the Uno are not spaced correctly between pins 7 and 8. There is a .160” gap between the pins. I got around this by bending pins 8-12 to fit. If I did this again, I would have used the proto shield (http://adafru.it/51), which accounts for this gap and makes it super easy to connect to the Uno.

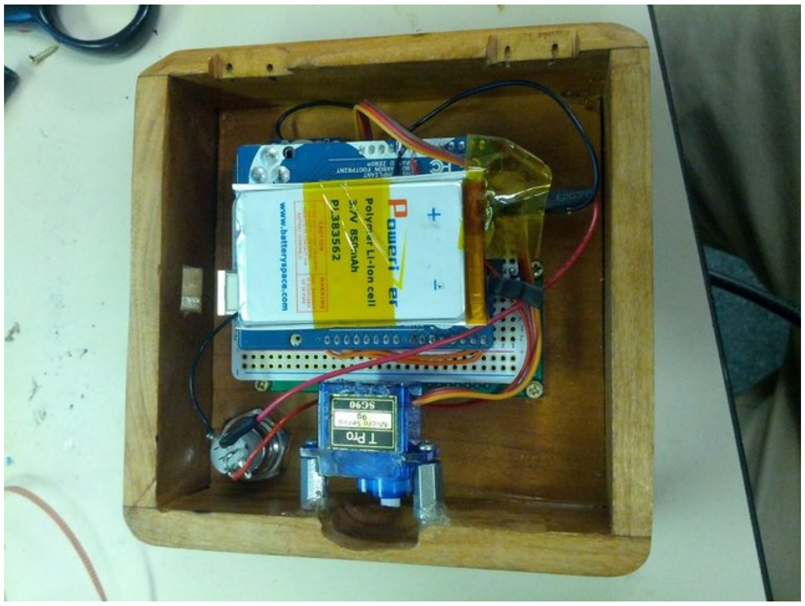
Enclosure

You can use pretty much any box you want for your reverse geocache. Make sure you don't cover up the GPS antenna with any metal that would block the reception. I made my box out of cherry and cut holes for the LCD and Power Button. I cut matching 45° angles on the back side to allow the box to open 90°.

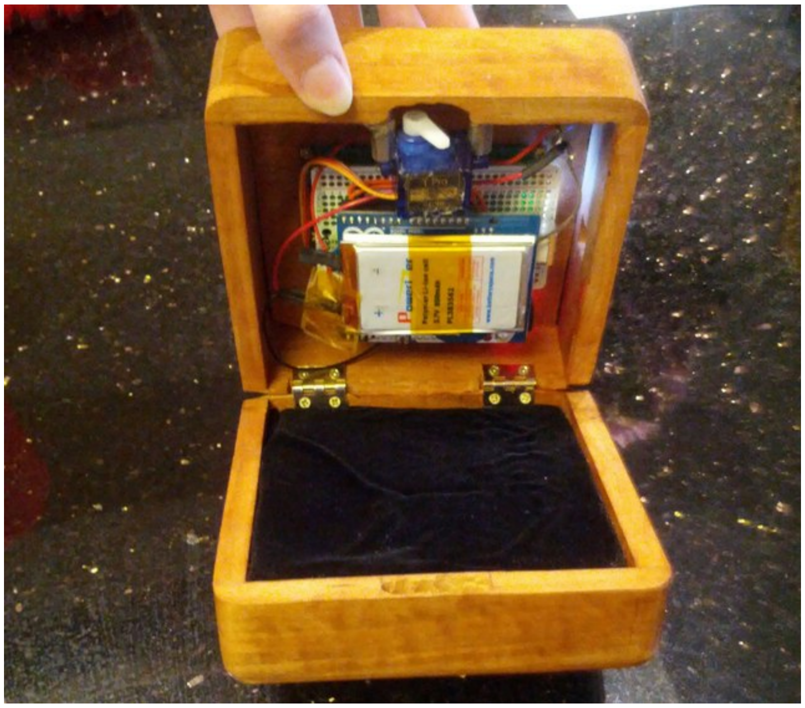




Next, insert all of your electronics. I cut a square hole in the side of the box with a jigsaw so I can connect to the Uno without taking it out of the box. I connected the circuit up to the power switch and mounted the servo on some standoffs.



I added some foam covered in velvet cloth on the bottom to hold the ring:



I added a latch mechanism to the servo arm that locks the box (not shown).

Software

I got a lot of help on the software from Bobby Nordlund who built his own Geobox that has a lot more features than mine. Check out his box here. (http://adafru.it/aTo) His stuff is based off of Mikal Hart of http://arduiniana.org (http://adafru.it/aTp) who had the original idea for the Reverse Geocache box. Below is the final version of the code I used:

```
/* Engagement Box   by Kenton Harris 11/12/2012
Reverse Geocache Engagement Ring Box
This program unlocks a box that has reached a certain location.

The device uses the following products from Adafruit Industries:
Arduino Uno: https://www.adafruit.com/products/50
Ultimate GPS (version1): http://www.adafruit.com/products/746
16x2 Character LCD: https://www.adafruit.com/products/181
TPro Micro Servo SG90: https://www.adafruit.com/products/169
Half Sized Perma proto: https://www.adafruit.com/products/571

Tutorials for these products found on learn.adafruit.com helped with much of this.
Copyright (c) 2012, Adafruit Industries
All rights reserved.

Thanks to bnordlund9 for much of the code. This is  simplified verison of his Geobox found here:
http://www.youtube.com/watch?v=g0060tcuofg
Credit to Mikal Hart of http://arduiniana.org/ for the original idea of Reverse Geocache.

*/

#include <math.h>
#include <LiquidCrystal.h>
#include <Adafruit_GPS.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3,2);
Adafruit_GPS GPS(&mySerial);
#define GPSECHO false            //make true to debug GPS
boolean usingInterrupt = false;
void useInterrupt(boolean);

//Servo
#include <PWMServo.h>
PWMServo servoLatch;

//Declarations
const float deg2rad = 0.01745329251994;
const float rEarth = 6371000.0;                         //can replace with 3958.75 mi, 6370.0 km
float range = 3000;                                   // distance from HERE to THERE
String here;                                        // read from GPS

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 8, 6, 10, 11, 12);

int gpsWasFixed = HIGH;                                 // did the GPS have a fix?
int ledFix = 4;                                     // pin for fix LED
int servoPin = 9;                                                    // pin for servo
int servoLock = 110;                                                   // angle (deg) of "locked" servo
int servoUnlock = 0;                                                   // angle (deg) of "unlocked" servo

String there = "N38 51.409, W077 01.328";                              //Desired Location goes here. Make

void setup()
{
  servoLatch.attach(SERVO_PIN_A);
  servoLatch.write(servoLock);
  delay(50);
  lcd.begin(16, 2);
  Serial.begin(115200);
  Serial.println("Debug GPS Test:");

  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);                        // RMC (recommended
  GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);                           // 1 Hz update rate
  useInterrupt(true);                                                  // reads the steaming data in a background
  delay(1000);


}

void loop(){
  // Parse GPS and recalculate RANGE
  if (GPS.newNMEAreceived()) {
    if (!GPS.parse(GPS.lastNMEA()))                                    // also sets the newNMEAreceived() flag t
      return;                                                          // We can fail to parse a sentence in which case w
  }
    if (GPS.fix) {
    gpsWasFixed = HIGH;
    digitalWrite(ledFix, HIGH);


    here = gps2string ((String) GPS.lat, GPS.latitude, (String) GPS.lon, GPS.longitude);
    range = haversine(string2lat(here), string2lon(here), string2lat(there), string2lon(there));
    Serial.print("Here: ");                                            //for GPS debug
    Serial.print(here);
    Serial.print("There: ");
    Serial.println(there);
    Serial.print("Range: ");
    Serial.print(range);
    Serial.println("m");

    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Distance to LOYL");
    //lcd.setCursor(0,1);
    //lcd.print("                ");
    lcd.setCursor(0,1);
    lcd.print(range);

    delay(500);
  }
  else {                                                               //No GPS fix- take box outside
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Hello Jenny!");
    lcd.setCursor(0,1);
    lcd.print("Take me outside!");
    delay(200);
  }

  if (range < 20.0){
    servoLatch.write(servoUnlock);
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Jenny: Will you ");
    lcd.setCursor(0,1);
    lcd.print("Marry Me?!");
    delay(5000);
  }
}


SIGNAL(TIMER0_COMPA_vect) {
  // Interrupt is called once a millisecond, looks for any new GPS data, and stores it
  char c = GPS.read();
  if (GPSECHO)
    if (c) UDR0 = c;
}

void useInterrupt(boolean v) {
  if (v) {
    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
    usingInterrupt = true;
  } else {
    TIMSK0 &= ~_BV(OCIE0A);
    usingInterrupt = false;
  }
}

String int2fw (int x, int n) {
  // returns a string of length n (fixed-width)
  String s = (String) x;
  while (s.length() < n) {
    s = "0" + s;
  }
  return s;
}

String gps2string (String lat, float latitude, String lon, float longitude) {
  // returns "Ndd mm.mmm, Wddd mm.mmm";
  int dd = (int) latitude/100;
  int mm = (int) latitude % 100;
  int mmm = (int) round(1000 * (latitude - floor(latitude)));
  String gps2lat = lat + int2fw(dd, 2) + " " + int2fw(mm, 2) + "." + int2fw(mmm, 3);
  dd = (int) longitude/100;
  mm = (int) longitude % 100;
  mmm = (int) round(1000 * (longitude - floor(longitude)));
  String gps2lon = lon + int2fw(dd, 3) + " " + int2fw(mm, 2) + "." + int2fw(mmm, 3);
  String myString = gps2lat + ", " + gps2lon;
  return myString;
```

```
};
/*
float string2radius (String myString) {
  // returns a floating-point number: e.g. String myString = "Radius: 005.1 NM";
  float r = ((myString.charAt(8) - '0') * 100.0) + ((myString.charAt(9) - '0') * 10.0) + ((myString.char
  return r;
};*/

float string2lat (String myString) {
  // returns radians: e.g. String myString = "N38 58.892, W076 29.177";
  float lat = ((myString.charAt(1) - '0') * 10.0) + (myString.charAt(2) - '0') * 1.0 + ((myString.charAt
  //Serial.print("float lat: ");
  //Serial.println(lat);
  lat *= deg2rad;
  if (myString.charAt(0) == 'S')
    lat *= -1;                                                  // Correct for hemisphere
  return lat;
};

float string2lon (String myString) {
  // returns radians: e.g. String myString = "N38 58.892, W076 29.177";
  float lon = ((myString.charAt(13) - '0') * 100.0) + ((myString.charAt(14) - '0') * 10.0) + (myString.c
  //Serial.print("float lon: ");
  //Serial.println(lon);
  lon *= deg2rad;
  if (myString.charAt(12) == 'W')
    lon *= -1;                                                  // Correct for hemisphere
  return lon;
};


float haversine (float lat1, float lon1, float lat2, float lon2) {
  // returns the great-circle distance between two points (radians) on a sphere
  float h = sq((sin((lat1 - lat2) / 2.0))) + (cos(lat1) * cos(lat2) * sq((sin((lon1 - lon2) / 2.0))));
  float d = 2.0 * rEarth * asin (sqrt(h));
  //Serial.println(d);
  return d;

};
```

When you turn on the box, it instructs the user to take it outside (to obtain the GPS satellites). Once it has a GPS fix, it displays the distance (in meters) to a programmed latitude/longitude location.

When it gets within 20 meters of the location, the servo rotates the latch, unlocking the box. The display then asks, “Will You Marry Me?”.

Support Forums

Support Forums (http://adafru.it/forums)