```
/*
  Reverse Geocache Box
 Based on Adafruit Reverse Geocache Box by Tyler Cooper
 Todd Miller July 2021
 This program unlocks a box that has reached a certain location.

 Notes: GPS RX -- Arduin pin 2  GPS TX -- Arduino pin 3
 LCD Pins 1,5, 16 to ground. 2, 15 to power 3 to potentiometer middle pin
 LCD 4 - Arduino 7
 LCD 6 - Arduino 8
 LCD 11 - Arduino 6
 LCD 12 - Arduino 10
 LCD 13 - Arduino 11
 LCD 14 - Arduino 12

Servo yellow wire to pin 9  brown wire to ground orange wire to +5v
*/



#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;
int SERVO_PIN_A = 9;

//Declarations
const float deg2rad = 0.01745329251994;
const float rEarth = 6371000.0;                    //can replace with 3958.75 mi, 6370.0 km, or
3440.06 NM
//float range = 3000;
float range = 804;                                  // distance from HERE to THERE
float yards = 3000;
String here;                                       // read from GPS

// initialize the lcd 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 = 105;                                  // angle (deg) of "locked" servo
int servoUnlock = 0;                                  // angle (deg) of "unlocked" servo


String there =  "N35 47.426, W078 38.083";  //Desired Location goes here. Make sure you use the same syntax and number of characters. Can be found on Google maps.
void setup()
{
  servoLatch.attach(SERVO_PIN_A);


  servoLatch.write(servoUnlock); // Allows 3 seconds to open box after turning it on for testing purposes. Remove this and delay below when ready.
  delay(3000);

  servoLatch.write(servoLock);
  delay(500);

  lcd.begin(16, 2);
  Serial.begin(115200);
  Serial.println("Debug GPS Test:");

  GPS.begin(9600);
  GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);                    // RMC (recommended minimum) and GGA (fix data) including altitude
  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 to false
      return;                                          // We can fail to parse a sentence in which case we should just wait for another
  }
  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(F("Here: "));                              //for GPS debug
    Serial.print(here);
```

```
    Serial.print(F("There: "));
    Serial.println(there);
    Serial.print(F("Range: "));
    Serial.print(range);
    Serial.println(F("m"));

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Meters Away"));
    lcd.setCursor(0, 1);
    lcd.print(range);


    delay(500);
  }
  else {                                        //No GPS fix- take box outside
   // lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Hello Betsy!"));
    lcd.setCursor(0, 1);
    lcd.print(F("Take me outside!"));
    delay(2000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Wait..."));
    lcd.setCursor(0, 1);
    lcd.print(F("Getting Signal"));
    delay(2000);
  }

  if (range < 100.0) { // Change this to how close you want someone to be to the target before box opens
in meters.  GPS probably not accurate below 3 or 4 meters.
    servoLatch.write(servoUnlock);
    delay(1000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(F("Betsy: You "));
    lcd.setCursor(0, 1);
    lcd.print(F("Have A Prize!"));
    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 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(4)
- '0') / 6.0) + ((myString.charAt(5) - '0') / 60.0) + ((myString.charAt(7) - '0') / 600.0) +
((myString.charAt(8) - '0') / 6000.0) + ((myString.charAt(9) - '0') / 60000.0);
  //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.charAt(15) - '0') * 1.0 + ((myString.charAt(17) - '0') / 6.0) + ((myString.charAt(18) - '0') /
60.0) + ((myString.charAt(20) - '0') / 600.0) + ((myString.charAt(21) - '0') / 6000.0) +
((myString.charAt(22) - '0') / 60000.0);
  //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;

};
```