GPS Dog Collar

Created by Becky Stern

https://learn.adafruit.com/gps-dog-collar

Last updated on 2025-01-22 12:12:36 PM EST

Table of Contents

Overview & Parts

Wonder if you're giving your dog the exercise she needs? Like electronics? Make your own GPS dog collar!

Parts

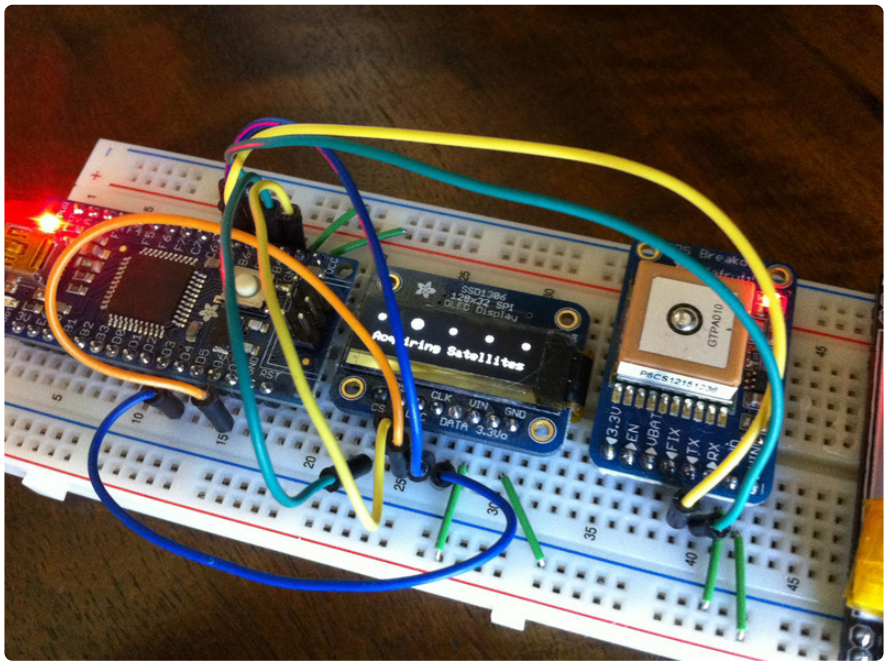
- Adafruit Ultimate GPS Breakout (http://adafru.it/746) - 66 channel w/10 Hz updates - MTK3339 chipset
- Atmega32u4 Breakout Board (http://adafru.it/296)
- Monochrome 128x32 OLED graphic display (http://adafru.it/661)
- 3 x AAA Battery Holder (http://adafru.it/727) and batteries
- stranded hookup wire
- wire-edged ribbon
- scrap of fabric
- electrical tape or dark nail polish (optional)

Tools:

- Soldering iron
- wire strippers
- pliers
- scissors
- needle and thread
- iron
- sewing pins
- sewing machine (optional)

The progress bar fills up as you stroll around the neighborhood. Set your goal ahead of time and watch the miles rack up.

Wiring



When Becky and I started this dog collar project, Arduino 1.0.1 had just come out with support for the Arduino Leonardo. I had an Adafruit 32u4 breakout board on hand, and thought it would be a good way to test it out with the official Leonardo bootloader. I used a USBtinyISP and the Arduino software to burn the Leonardo to the 32u4 breakout board without any issues.

My first step was getting the tiny, and awesome Adafruit Monochrome 128x32 OLED to work with the Leonardo software.

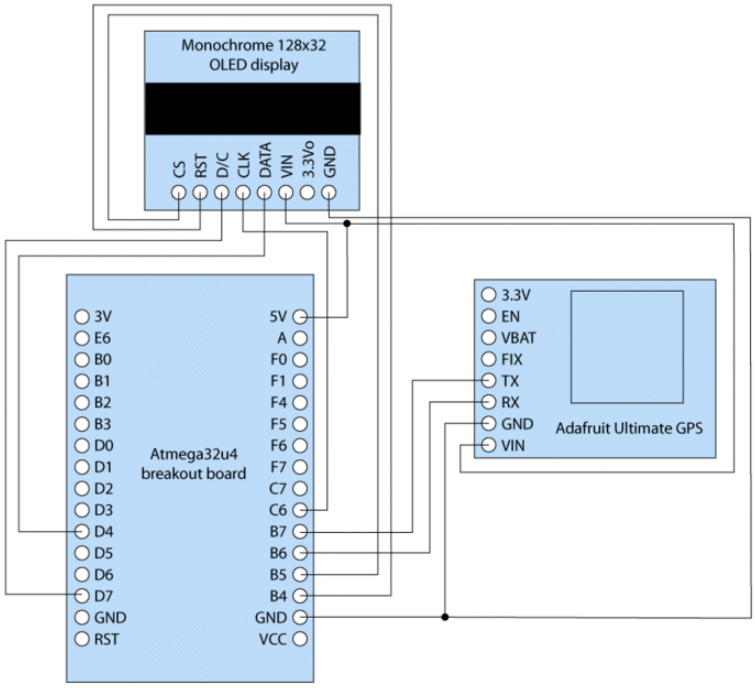
The only real issue I ran into was matching up the pins from the Atmega32u4 to the pins in the code examples/libraries for the OLED display. I ended up creating a pin reference sheet similar to this one by Johngineer (which is much better than mine). (https://adafru.it/aJf)

At the top of the code, I have comments showing what OLED pins match up to which pins on the 32u4 breakout.

```
#define OLED_DC 6     //Atmega32u4 Breakout Pin D7
#define OLED_CS 9     //Atmega32u4 Breakout Pin B5
#define OLED_CLK 5    //Atmega32u4 Breakout Pin C6
#define OLED_MOSI 4   //Atmega32u4 Breakout Pin D4 (Pin on OLED labeled DATA)
#define OLED_RESET 8  //Atmega32u4 Breakout Pin B4
```

I also show where to connect up the Adafruit Ultimate GPS module.

```
//Connect GPS TX to Atmega32u4 Breakout Pin B7 (Leonardo Pin D10)
//Connect GPS RX to Atmega32u4 Breakout Pin B6 (Leonardo Pin D11)
```



Code

The following is the code for the project.

```
// SPDX-FileCopyrightText: 2019 Anne Barela for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#define OLED_DC 6     //Atmega32u4 Breakout Pin D7
#define OLED_CS 9     //Atmega32u4 Breakout Pin B5
#define OLED_CLK 5    //Atmega32u4 Breakout Pin C6
#define OLED_MOSI 4   //Atmega32u4 Breakout Pin D4 (Pin on OLED labeled DATA)
#define OLED_RESET 8  //Atmega32u4 Breakout Pin B4

//Connect GPS TX to Atmega32u4 Breakout Pin B7 (Leonardo Pin D10)
//Connect GPS RX to Atmega32u4 Breakout Pin B6 (Leonardo Pin D11)

#include <SoftwareSerial.h>

#include <TinyGPS.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

/* This sample code demonstrates the normal use of a TinyGPS object.
   It requires the use of SoftwareSerial, and assumes that you have a
   4800-baud serial GPS device hooked up on pins 3(rx) and 4(tx).
```

```
*/

TinyGPS gps;
SoftwareSerial nss(11, 10);

static void gpsdump(TinyGPS &gps);
static bool feedgps();
static void print_float(float val, float invalid, int len, int prec);
static void print_int(unsigned long val, unsigned long invalid, int len);
static void print_date(TinyGPS &gps);
static void print_str(const char *str, int len);

//ENTER YOUR DESIRED DISTANCE GOAL (IN MILES)
//-------------------------------------------------------------------------------
float GOAL = 3; //Distances can include decimal points
//-------------------------------------------------------------------------------

// Maximum speed and last known position
float maxSpeed = 0;
float lastFlat = 0;
float lastFlon = 0;
long totalDistance = 0;

boolean start = 1;
int i = 0;

void setup()
{
  Serial.begin(9600);
  nss.begin(9600);

  // by default, we'll generate the high voltage from the 3.3v line internally!
(neat!)
  display.begin(SSD1306_SWITCHCAPVCC);
  // init done

  display.clearDisplay();
  display.display();
}

void loop()
{
  bool newdata = false;
  unsigned long start = millis();

  // Every second we print an update
  while (millis() - start < 1000)
  {
    if (feedgps())
      newdata = true;
  }

  gpsdump(gps);
}

static void gpsdump(TinyGPS &gps)
{
  float flat, flon;
  unsigned long age, date, time, chars = 0;
  unsigned short sentences = 0, failed = 0;

  gps.f_get_position(&flat, &flon, &age);

  //print_date(gps);

  //gps.stats(&chars, &sentences, &failed);
  //print_int(chars, 0xFFFFFFFF, 6);
  //print_int(sentences, 0xFFFFFFFF, 10);
  //print_int(failed, 0xFFFFFFFF, 9);
```

```
  //Serial.println();

  if (gps.f_speed_kmph() > 3.9)
  {
    if (start == 1)
    {
      start = 0;
      lastFlat = flat;
      lastFlon = flon;
    }
    else
    {
      //totalDistance = gps.distance_between(flat, flon, LONDON_LAT, LONDON_LON);
      totalDistance = totalDistance + calc_dist(flat, flon, lastFlat, lastFlon);
      lastFlat = flat;
      lastFlon = flon;
    }
  }

  display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);

  float fDist = totalDistance;
  //convert meters to miles
  fDist *= 0.000621371192;
  //float fSpeed = gps.f_speed_kmph();
  printLCDFloat(fDist, 2);
  display.print(" Miles (");

  float targetDist = fDist / GOAL;

  printLCDFloat(targetDist*100, 0);
  display.print("%)");

  display.drawLine(0, 12, 0, 31, WHITE);

  display.drawLine(63, 28, 63, 31, WHITE);
  display.drawLine(127, 12, 127, 31, WHITE);
  display.drawLine(31, 28, 31, 31, WHITE);

  display.drawLine(95, 28, 95, 31, WHITE);
  display.drawLine(0, 28, 127, 28, WHITE);
  display.drawLine(0, 12, 127, 12, WHITE);

  display.fillRect(2, 14, (124 * targetDist), 13, 1);

  if (gps.hdop() > 2000) {
    //display.fillRect(2, 14, (124), 13, BLACK);
    display.fillRect(0, 0, 128, 32, BLACK);
    display.fillCircle(6, 6, 2, WHITE);
    display.fillCircle(64, 6, 2, WHITE);
    display.fillCircle(122, 6, 2, WHITE);
    display.fillCircle(35, 6, 2, WHITE);
    display.fillCircle(93, 6, 2, WHITE);

    if (i==0){
      display.drawCircle(6, 6, 4, WHITE);
    }
    if (i==1){
      display.drawCircle(35, 6, 4, WHITE);
    }
    if (i==2){
      display.drawCircle(64, 6, 4, WHITE);
    }
    if (i==3){
      display.drawCircle(93, 6, 4, WHITE);
```

```
    }
    if (i==4){
      display.drawCircle(122, 6, 4, WHITE);
      i = 0;
    } else {
    i++;
    }

    display.setTextColor(WHITE);
    display.setCursor(5,20);
    display.print("Acquiring Satellites");
  }

  display.display();
}

static void print_int(unsigned long val, unsigned long invalid, int len)
{
  char sz[32];
  if (val == invalid)
    strcpy(sz, "*******");
  else
    sprintf(sz, "%ld", val);
  sz[len] = 0;
  for (int i=strlen(sz); i<len; ++i)
    sz[i] = ' ';
  if (len > 0)
    sz[len-1] = ' ';
  Serial.print(sz);
  feedgps();
}

static void print_float(float val, float invalid, int len, int prec)
{
  char sz[32];
  if (val == invalid)
  {
    strcpy(sz, "*******");
    sz[len] = 0;
        if (len > 0)
          sz[len-1] = ' ';
    for (int i=7; i<len; ++i)
        sz[i] = ' ';
    Serial.print(sz);
  }
  else
  {
    Serial.print(val, prec);
    int vi = abs((int)val);
    int flen = prec + (val < 0.0 ? 2 : 1);
    flen += vi >= 1000 ? 4 : vi >= 100 ? 3 : vi >= 10 ? 2 : 1;
    for (int i=flen; i<len; ++i)
      Serial.print(" ");
  }
  feedgps();
}

void printLCDFloat(double number, int digits)
{
  // Handle negative numbers
  if (number < 0.0)
  {
     display.print("-");
     number = -number;
  }

  // Round correctly so that print(1.999, 2) prints as "2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
```

```
    rounding /= 10.0;

  number += rounding;

  // Extract the integer part of the number and print it
  unsigned long int_part = (unsigned long)number;
  double remainder = number - (double)int_part;
  char sTemp[10];
  ltoa(int_part, sTemp, 10);
  display.print(sTemp);

  // Print the decimal point, but only if there are digits beyond
  if (digits > 0)
    display.print(".");

  // Extract digits from the remainder one at a time
  while (digits-- > 0)
  {
    remainder *= 10.0;
    int toPrint = int(remainder);
    ltoa(toPrint, sTemp, 10);
    display.print(sTemp);
    remainder -= toPrint;
  }
}

static void print_date(TinyGPS &gps)
{
  int year;
  byte month, day, hour, minute, second, hundredths;
  unsigned long age;
  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths,
&age);
  if (age == TinyGPS::GPS_INVALID_AGE)
    Serial.print("*******    *******    ");
  else
  {
    char sz[32];
    sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d   ",
        month, day, year, hour, minute, second);
    Serial.print(sz);
  }
  print_int(age, TinyGPS::GPS_INVALID_AGE, 5);
  feedgps();
}

static void print_str(const char *str, int len)
{
  int slen = strlen(str);
  for (int i=0; i<len; ++i)
    Serial.print(i<slen ? str[i] : ' ');
  feedgps();
}

static bool feedgps()
{
  while (nss.available())
  {
    if (gps.encode(nss.read()))
      return true;
  }
  return false;
}

unsigned long calc_dist(float flat1, float flon1, float flat2, float flon2)
{
  float dist_calc=0;
  float dist_calc2=0;
  float diflat=0;
```

```
  float diflon=0;

  diflat=radians(flat2-flat1);
  flat1=radians(flat1);
  flat2=radians(flat2);
  diflon=radians((flon2)-(flon1));

  dist_calc = (sin(diflat/2.0)*sin(diflat/2.0));
  dist_calc2= cos(flat1);
  dist_calc2*=cos(flat2);
  dist_calc2*=sin(diflon/2.0);
  dist_calc2*=sin(diflon/2.0);
  dist_calc +=dist_calc2;

  dist_calc=(2*atan2(sqrt(dist_calc),sqrt(1.0-dist_calc)));

  dist_calc*=6371000.0; //Converting to meters
  return dist_calc;
}
```

Code Walkthrough

If you scroll down a bit in the code, you will see where you enter the total distance you would like your dog to walk in a day. I have set the default to 3 miles.

```
//ENTER YOUR DESIRED DISTANCE GOAL (IN MILES)
//-----------------------------------------------------------------------------
float GOAL = 3; //Distances can include decimal points
//-----------------------------------------------------------------------------
```

I used the TinyGPS library (https://adafru.it/aJg) to do most of the heavy lifting, and pulled a lot of code from my Coobro Geo (http://adafru.it/652) code. The heart of the code is all about taking constant distance measurements. Every time the code loops, it looks at where you were, and where you are.

```
unsigned long calc_dist(float flat1, float flon1, float flat2, float flon2)
{
  float dist_calc=0;
  float dist_calc2=0;
  float diflat=0;
  float diflon=0;

  diflat=radians(flat2-flat1);
  flat1=radians(flat1);
  flat2=radians(flat2);
  diflon=radians((flon2)-(flon1));

  dist_calc = (sin(diflat/2.0)*sin(diflat/2.0));
  dist_calc2= cos(flat1);
  dist_calc2*=cos(flat2);
  dist_calc2*=sin(diflon/2.0);
  dist_calc2*=sin(diflon/2.0);
  dist_calc +=dist_calc2;

  dist_calc=(2*atan2(sqrt(dist_calc),sqrt(1.0-dist_calc)));

  dist_calc*=6371000.0; //Converting to meters
  return dist_calc;
}
```

When standing still, the GPS coordinates will jump around slightly. I didn't want this to affect the total distance traveled, so I had the code first make sure you were moving. If you are moving, it adds the distance value from the code above to a running total to determine your total distance traveled.

```
if (gps.f_speed_kmph() &gt; 3.9)
  {
    if (start == 1)
    {
      start = 0;
      lastFlat = flat;
      lastFlon = flon;
    }
    else
    {
      //totalDistance = gps.distance_between(flat, flon, LONDON_LAT, LONDON_LON);
      totalDistance = totalDistance + calc_dist(flat, flon, lastFlat, lastFlon);
      lastFlat = flat;
      lastFlon = flon;
    }
  }
```

My favorite part of coding this project was making that tiny OLED display useful information that would be easy to see and understand at a quick glance. I started by creating a neat 'Acquiring Satellites' animation when you first turn on the device. I then created a nice bar graph that shows your progress towards your goal. Above the bar graph is a running mileage counter. All of these numbers can easily be converted to km if needed.

```
display.clearDisplay();

  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,0);

  float fDist = totalDistance;
  //convert meters to miles
  fDist *= 0.000621371192;
  //float fSpeed = gps.f_speed_kmph();
  printLCDFloat(fDist, 2);
  display.print(" Miles (");

  float targetDist = fDist / GOAL;

  printLCDFloat(targetDist*100, 0);
  display.print("%)");

  display.drawLine(0, 12, 0, 31, WHITE);

  display.drawLine(63, 28, 63, 31, WHITE);
  display.drawLine(127, 12, 127, 31, WHITE);
  display.drawLine(31, 28, 31, 31, WHITE);

  display.drawLine(95, 28, 95, 31, WHITE);
  display.drawLine(0, 28, 127, 28, WHITE);
  display.drawLine(0, 12, 127, 12, WHITE);

  display.fillRect(2, 14, (124 * targetDist), 13, 1);

  if (gps.hdop() &gt; 2000) {
```

```
    //display.fillRect(2, 14, (124), 13, BLACK);
    display.fillRect(0, 0, 128, 32, BLACK);
    display.fillCircle(6, 6, 2, WHITE);
    display.fillCircle(64, 6, 2, WHITE);
    display.fillCircle(122, 6, 2, WHITE);
    display.fillCircle(35, 6, 2, WHITE);
    display.fillCircle(93, 6, 2, WHITE);

    if (i==0){
      display.drawCircle(6, 6, 4, WHITE);
    }
    if (i==1){
      display.drawCircle(35, 6, 4, WHITE);
    }
    if (i==2){
      display.drawCircle(64, 6, 4, WHITE);
    }
    if (i==3){
      display.drawCircle(93, 6, 4, WHITE);
    }
    if (i==4){
      display.drawCircle(122, 6, 4, WHITE);
      i = 0;
    } else {
    i++;
    }

    display.setTextColor(WHITE);
    display.setCursor(5,20);
    display.print("Acquiring Satellites");
  }

  display.display();
```

Going Further

The Original Summary

That about sums up the main parts of the code. Your challenge, should you choose to accept, is to improve upon what we have done here. The next logical step would be to use the built in data logger feature of the Ultimate GPS module, and map your dog walks when you get home.

Going Further

This project was completed in 2012. Advancements in parts are such that this project might be more easily done on a board such as a Feather M0 in CircuitPython.

As far as replicating the original project: Adafruit now carries the original ATmega32u4 processor on the Feather 32u4 Basic and Feather 32u4 Adalogger boards. The latter provides an SD card capability good for data logging. At present we do not have diagrams and code to implement the project with these boards but the changes from the original processor board would be very small. The boards are listed in the Featured Products page.