SCI Communications & GPS

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #25

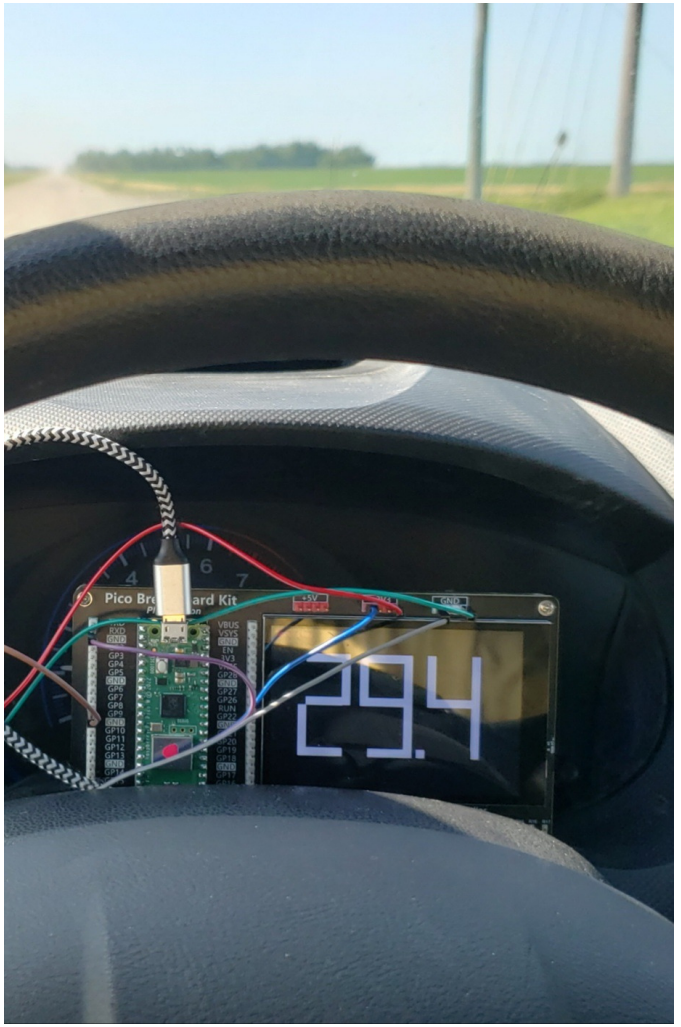
Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

Another sensor available for use with a Pi-Pico is a GPS sensor.

These tell you your

- Latitude
- Longitude
- Elevation
- Speed in knots, and
- Heading in degrees



With a GPS sensor, you can

- Locate your car
  - Report distance from a reference position
- Display your speed in mph
- Other (up to your creativity)

SCI Communication

- Serial Communications Interface

SCI is a type of asynchronous communications.

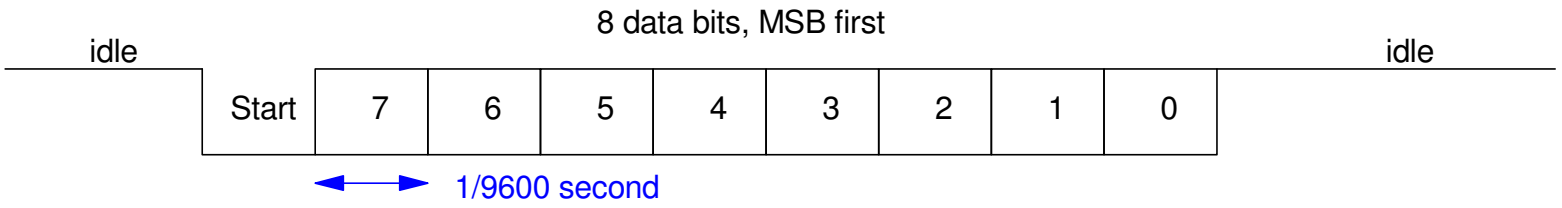
- No clock

Data is sent in 8-bit packets

- Start Bit
- 8 Data Bits
- 0-3 Stop Bits

Each bit is a fixed width

- 1/9600 second for 9600 baud



SCI Data: Start bit comes first

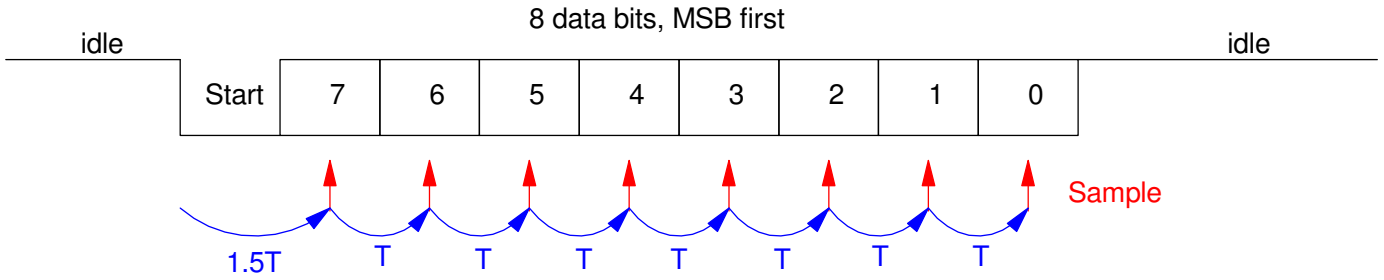
Receiving Serial Data

When receiving serial data,

- The falling edge indicates the start of a message
- Sample each bit (blindly) in the middle of each bit
- After eight reads, you have one byte

This really need to be done using interrupts or hardware

- Timing is critical



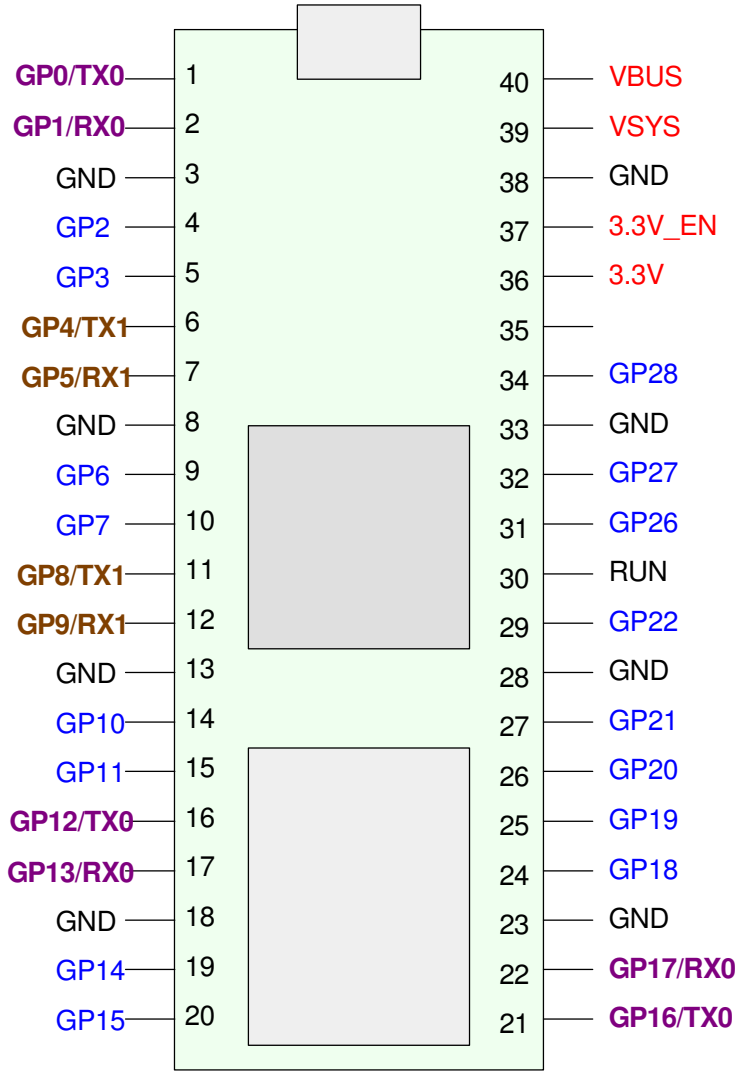
GPS Read: Sample each bit in the middle to determine its value.

SCI Ports on a Pi-Pico

The Pi-Pico has two SCI ports available:

- SCI0:
  - TX=GP0, RX=GP1, or
  - TX=GP12, RX=GP13, or
  - TX=GP16, RX=GP17
- SCI1
  - TX=GP4, RX=GP5, or
  - TX=GP8, RX=GP9

Your pick which port and pins you use



The procedure to initialize a SCI port is

```
from machine import UART

uart = UART(0, 9600)
uart.init(9600, bits=8, parity=None, stop=1, tx=0, rx=1)
```

Different ways to read and write to a UART are:

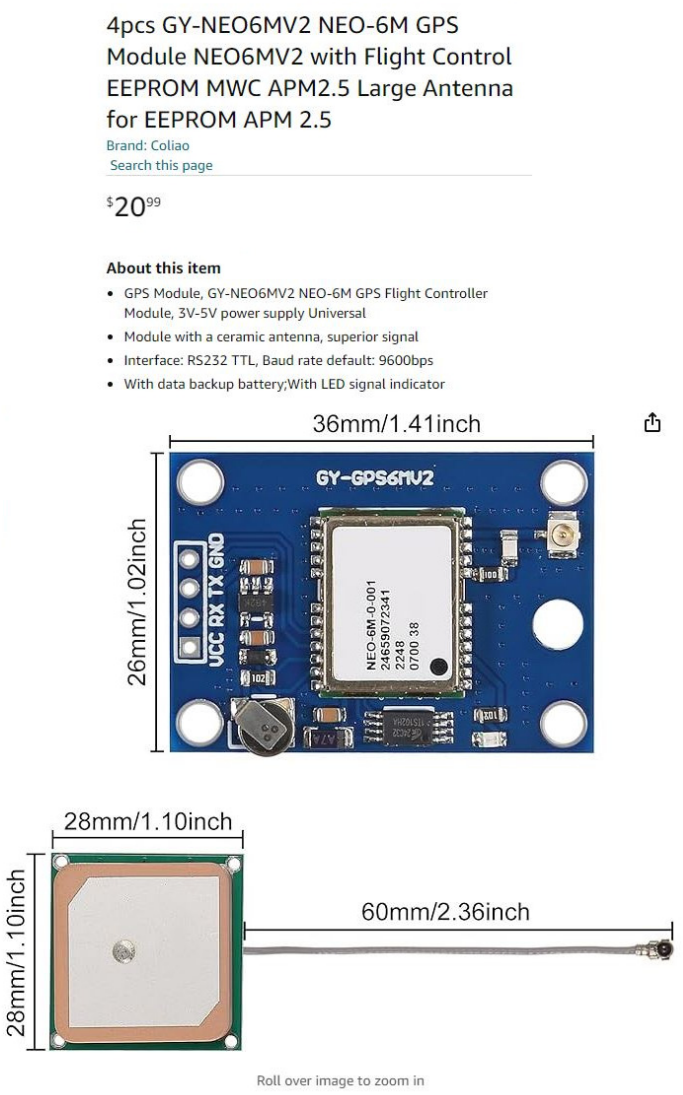
```
uart.read(5)            # read 5 characters into a buffer
uart.read()             # read all available characters
uart.readline()         # read a line (stop at carriage return)
uart.readinto(buf)      # read and store in a buffer
uart.write('Hello')     # write to the SCI port
```

GPS Modules & Messages

GPS sensors use satellites to tell you your location.

They're actually really easy to use:

- Connect power and ground
- Connect the serial out (TX) to the serial in (RX) on the Pi-Pico
- Receive serial data at 9600 baud (default).

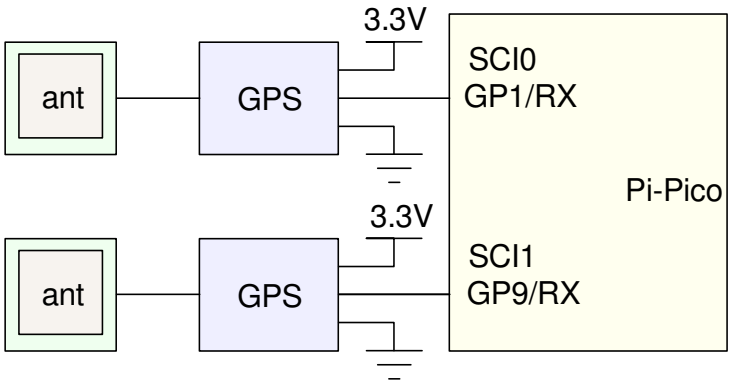


Hardware:

Up to two GPS sensors can be read

- SCI0
- SCI1

Only one wire is needed if you stick with 9600 baud



Reading GPS Messages

- Initialize the serial port to 9600 baud
- Read each message

```
from machine import UART
from time import sleep

uart = UART(1, 9600)
uart.init(9600, bits=8, parity=None, stop=1, tx=8, rx=9)

while(1):
    x = uart.readline()
    sleep(0.2)
```

shell

```
b'$GPGGA,205246.00,4649.55240,N,09652.11367,W,1,07,1.17,283.7,M,-27.5,M,,*69\r\n'
b'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,3.59,1.17,3.39*0C\r\n'
b'$GPGSV,2,1,08,08,19,311,09,10,52,288,24,15,28,055,21,18,47,147,25*78\r\n'
b'$GPGSV,2,2,08,23,77,015,19,24,39,100,21,27,32,277,1
b'$GPRMC,205247.00,A,4649.55258,N,09652.11395,W,0.306,,140724,,,A*62\r\n'
b'$GPVTG,,T,,M,0.306,N,0.567,K,A*22\r\n'
b'$GPGGA,205247.00,4649.55258,N,09652.11395,W,1,07,1.14,284.1,M,-27.5,M,,*6E\r\n'
b'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,2.49,1.14,2.22*04\r\n'
b'$GPGSV,2,1,08,08,19,311,08,10,52,288,25,15,28,055,22,18,47,147,26*78\r\n'
b'$GPGSV,2,2,08,23,77,015,19,24,39,100,21,27,32,277,1
b'$GPRMC,205248.00,A,4649.55297,N,09652.11403,W,0.312,,140724,,,A*63\r\n'
b'$GPVTG,,T,,M,0.312,N,0.578,K,A*29\r\n'
b'$GPGGA,205248.00,4649.55297,N,09652.11403,W,1,07,1.14,284.5,M,-27.5,M,,*6E\r\n'
b'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,2.49,1.14,2.22*04\r\n'
```

GPS Message: $GPGGA

- **$GPGGA**,205246.00,4649.55240,N,09652.11367,W,1,07,1.17,283.7,M,-27.5,M,,*69
- UTC time (hhmmss.sss).
  - Data was recorded at 20:42:46.00 seconds GMT
- Latitude (ddmm.mmmm).
  - 46 degrees, 49.55240 minutes north
- Longitude (ddmm.mmmm)
  - 096 degrees 42.11367 minutes west
- Fix
  - 0: Fix not available or invalid
  - 1: GPS SPS mode, fixed valid
  - 2: Differential GPS, SPS mode, fix valid
- Satellites used (07, 1.17)
- Altitude (283.7 meters)

$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,3.59,1.17,3.39*0C

- Satellites used in the position solution
- {3, 23, 18, 10, 27, 15, 32, 24}

$GPGSV,2,1,08,08,19,311,09,10,52,288,24,15,28,055,21,18,47,147,25*78

- The number of satellites in view

$GPRMC,205247.00,A,4649.55258,N,09652.11395,W,0.306,,140724,,,A*62

- Time, Data, Position, Course, and Speed
- A = valid data, V = invalid data
- Time (hhmmss.ss). Current time is 20:52:47.00
- Latitude (ddmm.mmmm). Location is 46 deg 49.55258 minutes north
- Longitude (ddmm.mmmm) Location is 096 deg 52.11395 minutes west
- Speed in knots: Speed is 0.306 knots
- Direction (in degrees)

Reading in a GPS Message

The Python command *uart.readline()* works in theory

- Seems to be inconsistent
- Misses some carriage returns
- Doesn't start with a $

So, write custom routines

- Use bottom-up programming

```
b'$GPGGA,205246.00,4649.55240,N,09652.11367,W,1,07,1.17,283.7,M,-27.5,M,,*69\r\n'
b'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,3.59,1.17,3.39*0C\r\n'
b'$GPGSV,2,1,08,08,19,311,09,10,52,288,24,15,28,055,21,18,47,147,25*78\r\n'
b'$GPGSV,2,2,08,23,77,015,19,24,39,100,21,27,32,277,1
b'$GPRMC,205247.00,A,4649.55258,N,09652.11395,W,0.306,,140724,,,A*62\r\n'
b'$GPVTG,,T,,M,0.306,N,0.567,K,A*22\r\n'
b'$GPGGA,205247.00,4649.55258,N,09652.11395,W,1,07,1.14,284.1,M,-27.5,M,,*6E\r\n'
b'$GPGSA,A,3,23,18,10,27,15,32,24,,,,,,2.49,1.14,2.22*04\r\n'
b'$GPGSV,2,1,08,08,19,311,08,10,52,288,25,15,28,055,22,18,47,147,26*78\r\n'
```

Level 1:

- **GPS_Read_Line(chan):**

Supports two GPS units

- SCI0 & SCI1

Read from the serial port

- Go character by character
- $ indicated start of message
- Carriage return (13) indicates end of message
- Return a string of everything in-between

Use a flag

- Keep reading until you see a carriage return

```
def GPS_Read_Line(chan):
    flag = 0
    n = 0
    msg = ''
    while(flag == 0):
        if(chan == 0):
            x = uart0.read(1)
        else:
            x = uart1.read(1)
        if(x != None):
            x = ord(x)
            if(chr(x) == '$'):
                msg = ''
            if(x == 13):
                flag = 1
            else:
                msg = msg + chr(x)
    return(msg)
```

Test Code:

- Check that GPS_Read_Line(chan) works

```
while(1):
    msg = GPS_Read_Line(0)
    print(msg)
```

shell

```
$GPVTG,,T,,M,0.970,N,1.797,K,A*25
$GPGGA,173924.00,4649.55763,N,09652.11931,W,1,06,1.27,288.7,M,-27.5,M,,*60
$GPGSA,A,3,29,18,15,13,20,23,,,,,,,2.52,1.27,2.18*0F
$GPGSV,4,1,16,01,21,283,17,05,55,055,07,07,02,027,,11,08,099,*7C
$GPGSV,4,2,16,13,42,105,19,15,45,156,12,16,11,324,,18,50,296,19*7F
$GPGSV,4,3,16,20,26,062,14,23,20,238,20,25,00,206,,26,16,289,*75
$GPGSV,4,4,16,29,64,190,19,30,03,056,,46,28,221,,48,30,216,*77
$GPGLL,4649.55763,N,09652.11931,W,173924.00,A,A*7F
$GPRMC,173925.00,A,4649.55729,N,09652.11947,W,1.143,,180724,,,A*67
$GPVTG,,T,,M,1.143,N,2.116,K,A*20
```

Level 2: String_to_Num()

Convert fields to numbers

- Without crashing if the field isn't a valid number

Fields are in fixed locations

- Example: GPRMC message

```
x = '$GPRMC,173925.00,A,4649.55729,N,09652.11947,W,1.143,,180724,,,'
```

The fields can be pulled out as:

$GPRMC,	173925.00	46	49.55729	096	52.11947	1.143
location	x[7:16]	x[19:21]	x[21:29]	x[32:35]	x[35:43]	x[46:51]
meaning	hhmmss GMT	latitude degrees	latitude minutes	longitude degrees	longitude minutes	speed knots

Str2Num(x)

float(x) doesn't work

- crashes program if x isn't a valid number string

This program

- Pulls out digits
- Allows decimal places
- Doesn't crash if field is wrong
  - just sets Error_Flag

```
def Str2Num(X):
    global Error_Flag
    n = len(X)
    y = 0
    flag = 0
    k = 0
    for i in range(0,n):
        z = X[i]
        if(z in {'0','1','2','3',...):
            if(z == '.'):
                flag = 1
            else:
                if(flag == 0):
                    y = 10*y + int(z)
                else:
                    k -= 1
                    y += int(z)*(10**k)
        else:
            Error_Flag = 1
    return(y)
```

Testing Str2Num

For a valid number:

```
Error_Flag = 0
msg = '123.456'
print(Str2Num(msg), Error_Flag)
```

shell

```
123.456   0
```

For an invalid number:

```
Error_Flag = 0
msg = '1G3.456'
print(Str2Num(msg), Error_Flag)
```

shell

```
13.456   1
```

Level 3: GPS_Read(chan)

Read in a GPS message

Keep reading until GPRMC is found

- character 3 is 'R'
- Message length > 52

Pull out each field

- time
- Latitude
  - degrees
  - minutes
- Longitude
  - degrees
  - minutes
- Speed

Return reading

- varies in following programs

```
def GPS_Read(chan):
  flag = 0

  while(flag == 0):
    x = GPS_Read_Line(chan)
    if(len(x) > 52):
      if(x[3] == 'R'): # $GPRMC
        flag = 1
        time = Str2Num(x[7:16])
        LatD = Str2Num(x[19:21])
        LatM = Str2Num(x[21:29])
        LonD = Str2Num(x[32:35])
        LonM = Str2Num(x[35:43])
        speed = Str2Num(x[46:51])

  return([time, LatD, LatM, LonD,
LonM, speed])
```

Test Routine

- Read the GPS over and over
- Display time, latititude, longitude, and speed
  - Time increments by one (no messages are missed)
  - Latitude and longitude is Fargo, ND (correct)
  - Speed is zero-ish
- Everything looks good

```
while(1):
    [t, xd, xm, yd, ymv] = GPS_Read(0)
    msg0 = str('{:7.0f}'.format(t) + ' ')
    msg1 = str('{:11.7f}'.format(xd+xm/60) + ' ')
    msg2 = str('{:11.7f}'.format(yd+ym/60) + ' ')
    msg3 = str('{:9.4f}'.format(v) + ' ')
    print(msg0 + msg1 + msg2 + msg3)
```

shell

```
183300  46.8258247  96.8686447    0.1620
183301  46.8258247  96.8686447    0.1150
183302  46.8258286  96.8686447    0.0290
183303  46.8258286  96.8686447    0.0490
```

Where's My Car?

- Video: Where's my cat?

Level 4: Main Routine

- *a fairly long routine*
- *Full code posted on Bison Academy*

Displays

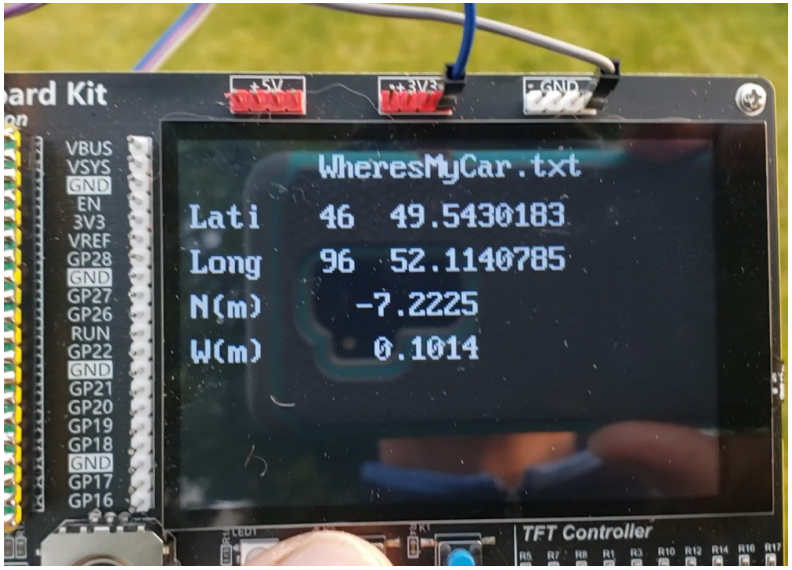
- Current GPS Location
- Distance to marked location

GP15: Mark Current Location

- Use current location as (0,0) position

GP14: Record Data

- Toggle recording on/off
- beep: recording & file append
- beep-beep: recording turned off & file closed



Data File:

The data file contains six columns of number

- Latitude in degrees & minutes
- Longitude in degrees & minutes
- Distance north of your home position in meters
- Distance west of your home position in meters

```
Latutude         Longitude      North   West
deg min         deg min         (m)     (m)
46 49.5489616  49 52.1171951  0.6142  0.4879
46 49.5489616  49 52.1171951  0.6142  0.4879
46 49.5490990  49 52.1174164  0.8684  0.7680
46 49.5489388  49 52.1175575  0.5719  0.9468
```

Text file for Wheres_My_Car.py. GPS position and distance to home position in meters

Unit Conversions

Longitude (E/W):

- The Earth's equatorial circumference is 40,075km (space.com).
- At the equator, each degree is 111.317km

$$1^0 = \frac{40{,}075km}{360} = 111.319km$$

- Each minute is 1855.285m

$$1' = \frac{40{,}075km}{60 \cdot 360} = 1{,}855.324m$$

Scale by your latitude (assume 46.8258 degrees north)

$$1' = \left(\frac{40{,}075km}{60 \cdot 360}\right) \cdot \cos(46.8258^0) = 1{,}269.448m$$

So, in Fargo, one minute of longitude corresponds to 1269.448 meters east/west.

Unit Conversions

Latitude:

- The Earth's polar circumference is 40,008km (space.com).
- Each degree of latitude corresponds to

$$1^0 = \frac{40{,}008km}{360} = 111.133km$$

Each minute of latitude corresponds to 1852.222 meters

$$1^{'} = \frac{40{,}008km}{60 \cdot 360} = 1852.222m$$

With these conversions,

- If you know your distance from home in minutes
- You know your distance in meters

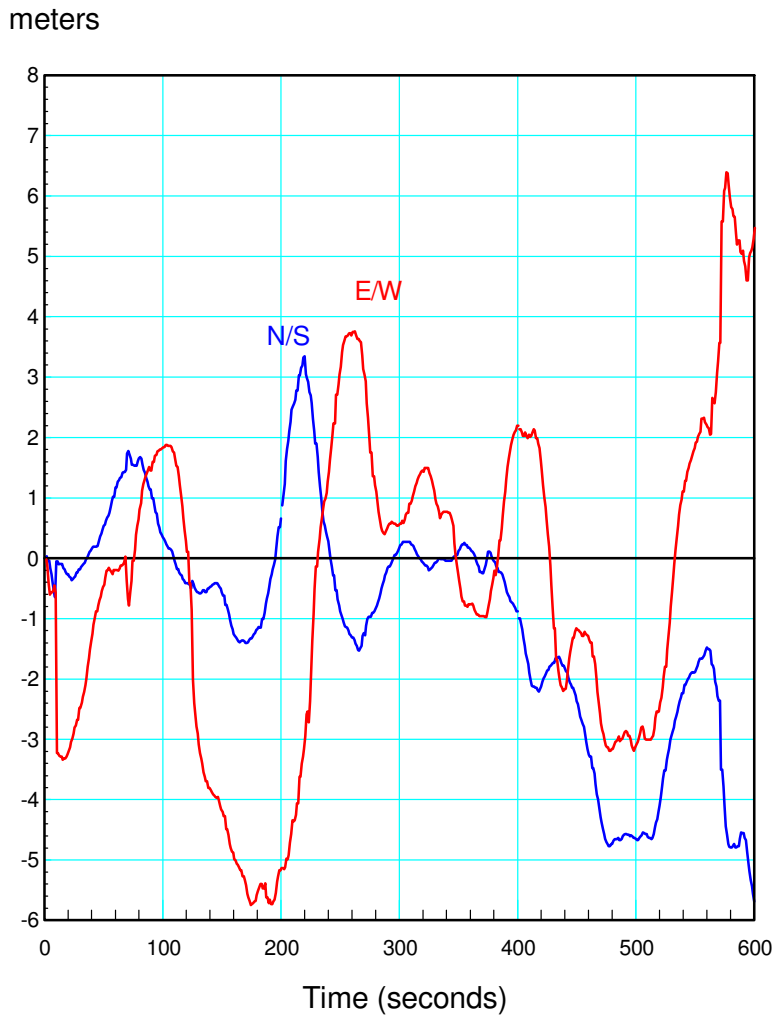
Noise on GPS Signals

The GPS readings drift

- Atmospheric disturbances
- Calculation errors in GPS module
  - Using a $5 GPS module
  - More expensive ones exist

Example: GPS location over 10 minutes

- Sensor is stationary
- Reported position is drifting



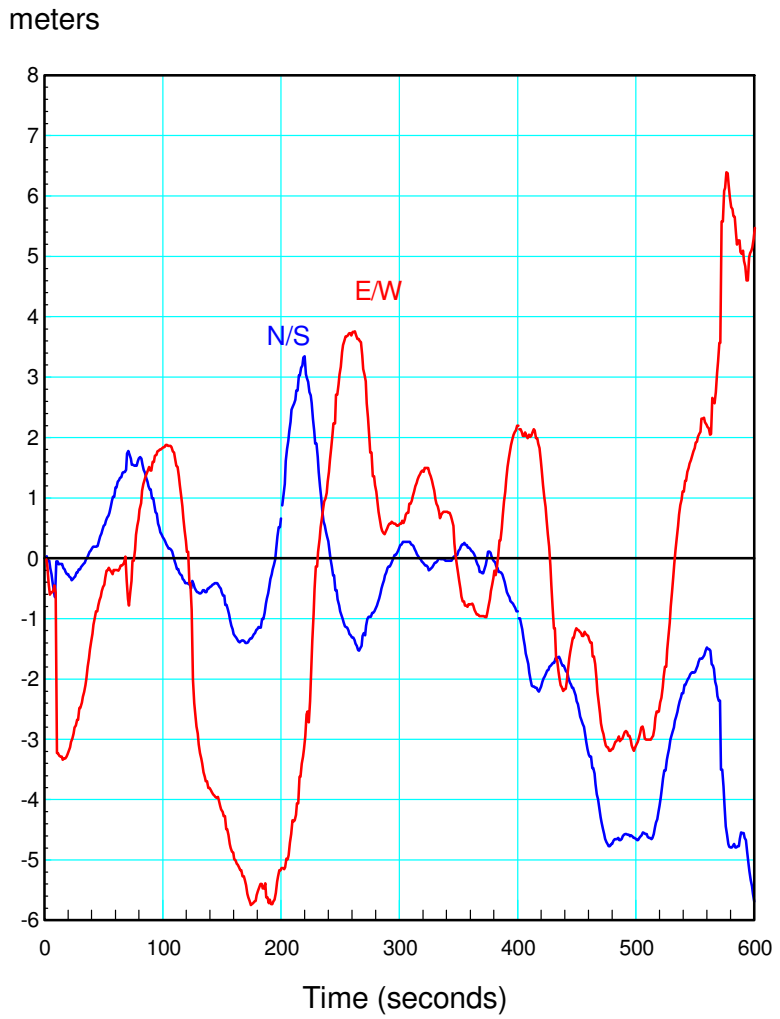
GPS Noise: Statistics

90% confidence interval

- N/S = +/- 3.15 meters
- E/W = +/- 4.57 meters

With this sensor, you know your position within 5 meters (ish)

	mean	st dev	1.66 * st dev
N/S	-1.025 m	1.897 m	3.15 m
E/W	-0.436 m	2.751 m	4.57 m



GPS Noise When Moving

Walk around a rectangle

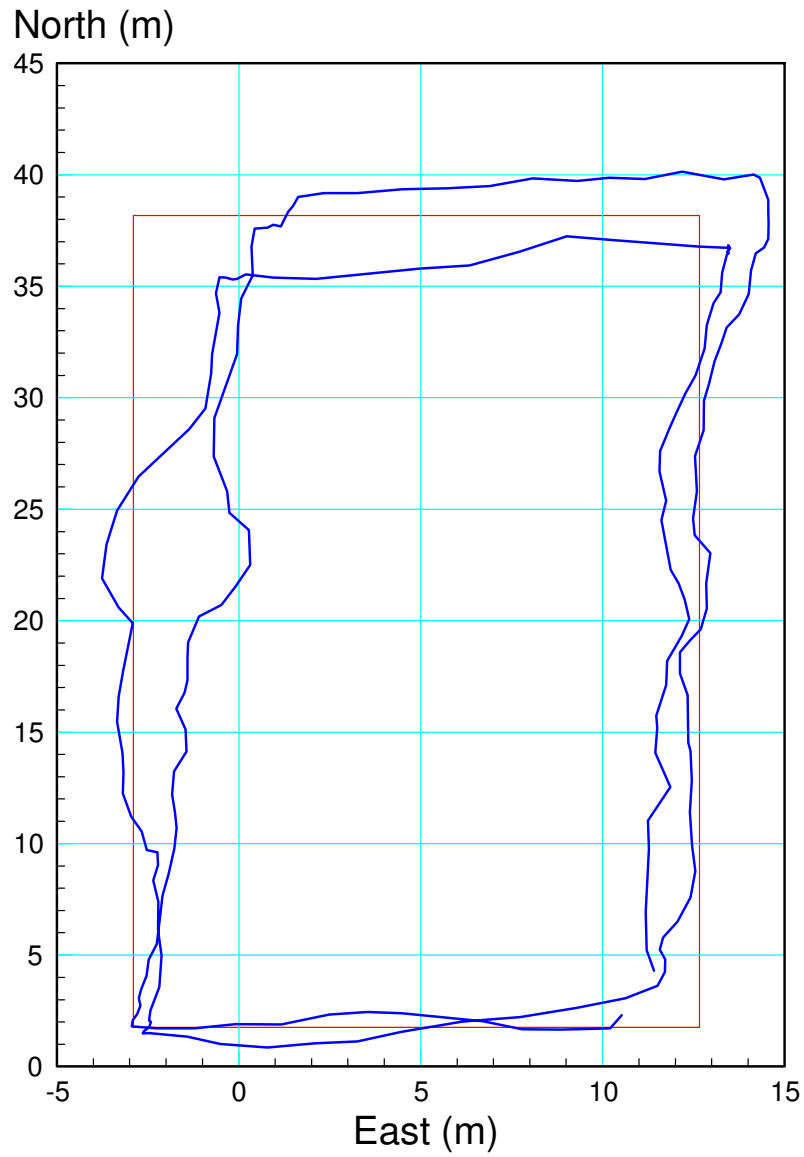
- shown in red

Record GPS position

- shown in blue

Tracking is a little better when moving

- +/- 2 meters (?)
- Hard to say without a known position



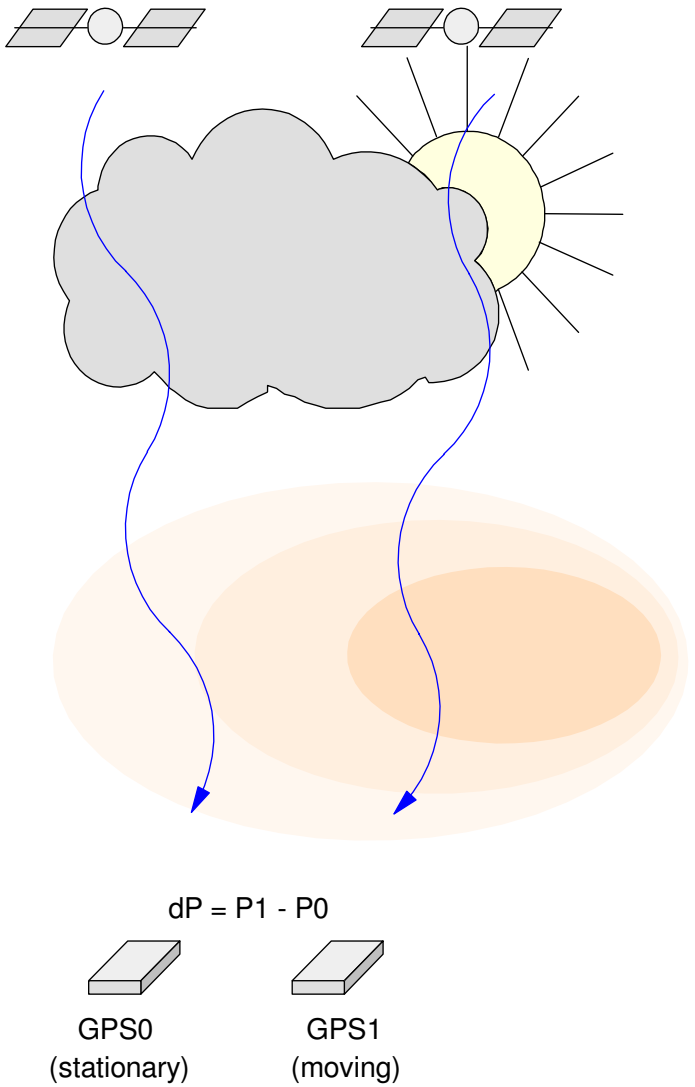
Differential GPS

One way tpo improve GPS accuracy

- Assume disturbances on GPS readings are caused by atmospheric disturbances etc.
- Assume two GPS units close to each other will have the same drift

Taking the difference in GPS readings *should* cause the disturbances to cancel

- Net result is improved GPS accuracy



Differential GPS with a Pi-Pico

Use two GPS receivers

- One of SCI0
- One on SCI1

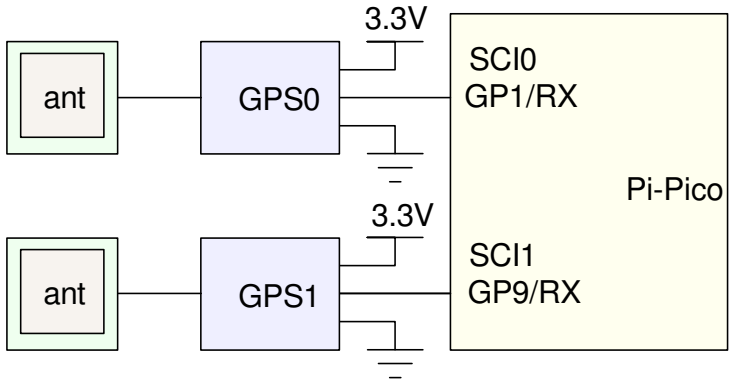
One GPS has a known position

- Fixed position

The other GPS can move around

The difference in GPS readings *should* have lower drift

- Smaller standard deviation in readings



Differential GPS Code:

Read both GPS receivers

- Keep reading until you get a valid GPS signal from both

Use the difference in readings

- x = x1 - x0
- Sensor minus reference

```
while(1):
    Error_Flag = 1
    while(Error_Flag == 1):
        Error_Flag = 0
        [t0, x0, y0, v0] = GPS_Read(0)
        [t1, x1, y1, v1] = GPS_Read(1)
    x = x1 - x0
    y = y1 - y0
```

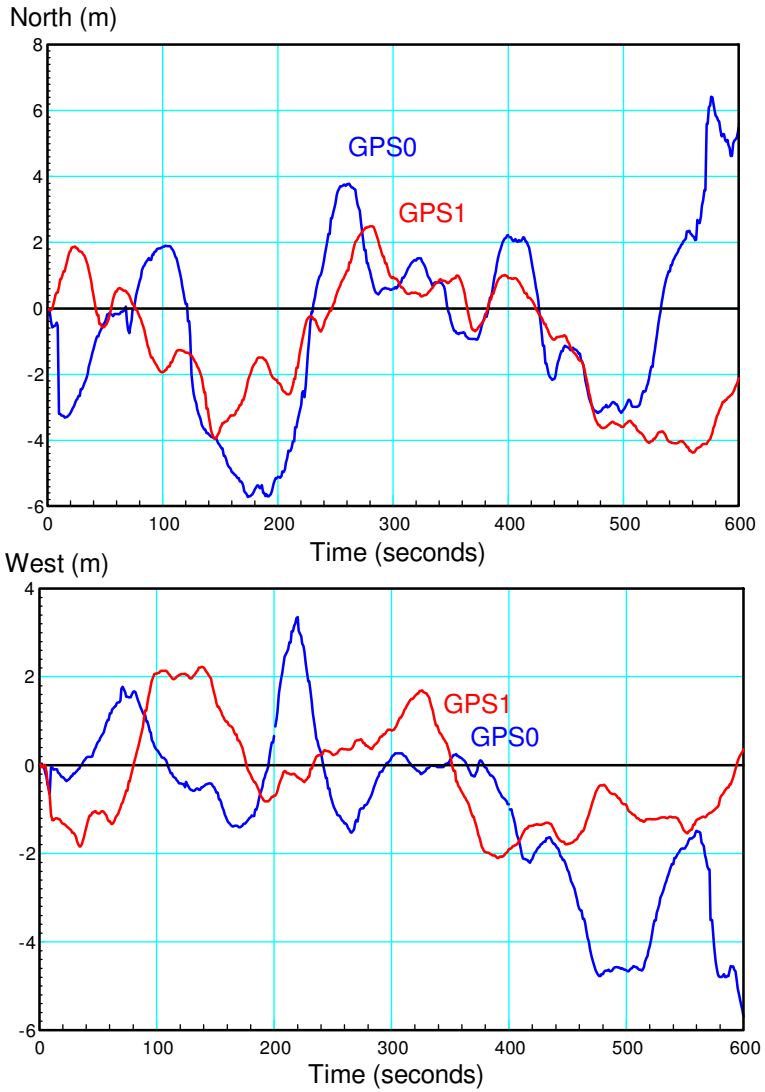
Differential GPS Results

- Both sensors stationary
- 20cm apart
- N/S position (top)
- E/W position (bottom)

Observations:

- Not much correlation in graphs
- Subtracting actually makes noise worse
  - Standard deviation increases
  - Supposed to decrese :(

	std(S0)	std(S1)	std(S0-S1)
N/S	2.7507 m	1.9012 m	2.9768 m
E/W	1.8974 m	1.1917 m	1.9349 m



Correlation of GPS Readings:

Calculate the correlation between each GPS sensor

- ECE 341 lecture #19

A correlation of 1.00 is good

- Same noise on both signals
- Noise will cancel

A correlation of 0.00 is bad

- Uncorrelated noise on two sensors
- Noise does not cancel
- Subtracting actually increases noise

$$cov(x, y) = E(xy) - E(x)E(y)$$

$$\rho_{x,y} = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

Matlab Code

```
>> num = mean(x0 .* x1) -
mean(x0)*mean(x1);

>> den = std(x0) * std(x1);

>> rhox = num / den

rhox =    0.2214

>> num = mean(y0 .* y1) -
mean(y0)*mean(y1);

>> den = std(y0) * std(y1);

>> rhoy = num / den

rhoy =    0.2817
```

Summary: Differential GPS

Correlation Coefficient

- N/S: 0.2817
- E/W: 0.2214

Very weak correlation

- Noise on the two sensors is mostly uncorrelated
- Noise does not have a common source
- Differential GPS doesn't work with the GPS sensors used
  - $5 GPS sensor
- Probably need a more expensive GPS units for differential GPS to work

GPS Speedometer

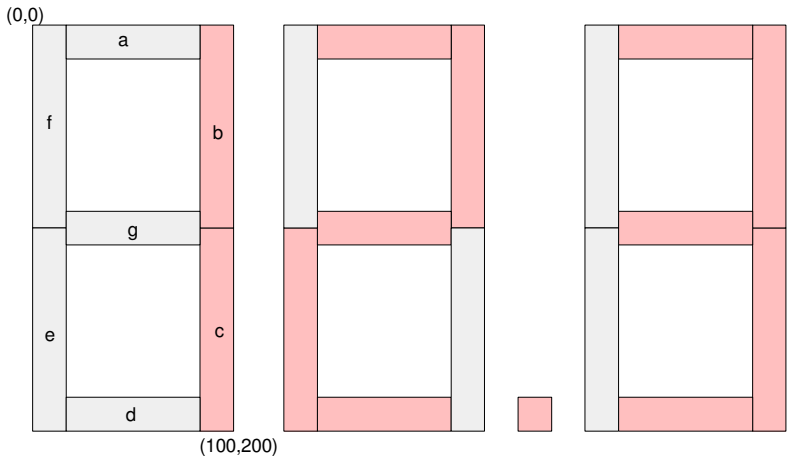
Finally, let's use the GPS sensor to measure your speed in mph.

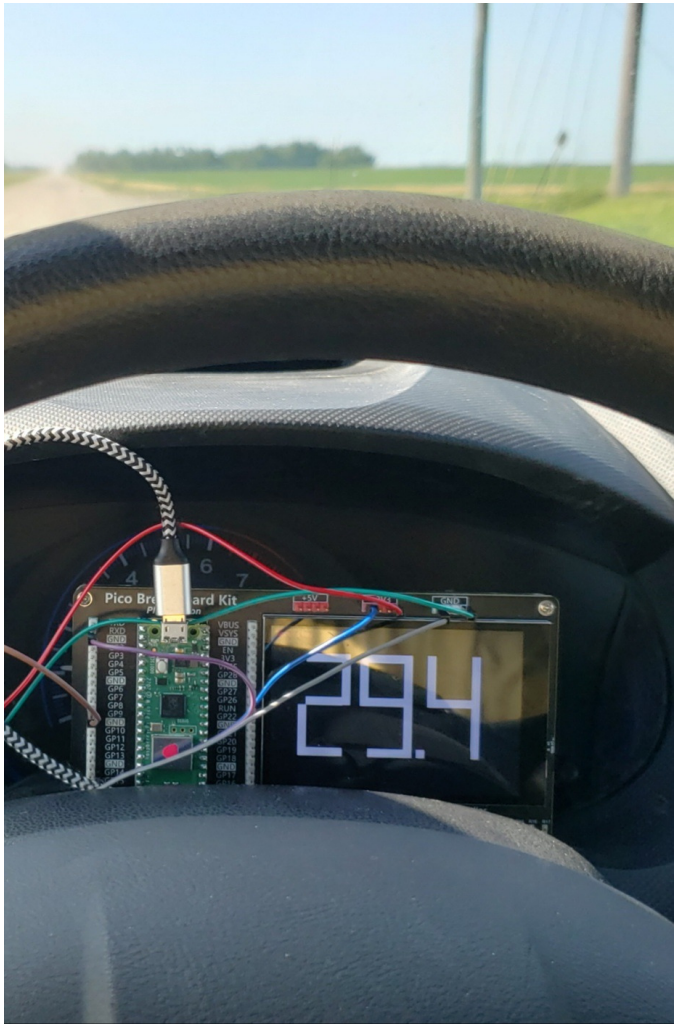
$GPRMC message includes speed in knots

$$mph = knots \cdot 1.15077$$

Display the speed in 100 x 200 font

- Big numbers so the driver can see them





Big_Display()

Bottom-Up Programming

- Lowest level

Display a single digit

- 0 - 9
- 100 pixels wide
- 200 pixels high

Use seven boxes for each number

- Similar to 70segment display
- White to light up
- Black for dark

```
def Big_Display(N, x, y):
  T = 15
  c1 = LCD.RGB(250,250,250)
  c0 = 0
  if(N == 0):
    LCD.Solid_Box(x+T,y,x+100-T,y+T,c1)
    LCD.Solid_Box(x+100-T,y,x+100,y+100,c1)
    LCD.Solid_Box(x+100-T,y+100,x+100,y+200,c1)
    LCD.Solid_Box(x+T,y+200-T,x+100-T,y+200,c1)
    LCD.Solid_Box(x,y+100,x+T,y+200,c1)
    LCD.Solid_Box(x,y,x+T,y+100,c1)
    LCD.Solid_Box(x+T,y+100-T,x+100-T,y+100,0)
  if(N == 2):
    LCD.Solid_Box(x+T,y,x+100-T,y+T,c1)
    LCD.Solid_Box(x+100-T,y,x+100,y+100,c1)
    LCD.Solid_Box(x+100-T,y+100,x+100,y+200,0)
    LCD.Solid_Box(x+T,y+200-T,x+100-T,y+200,c1)
    LCD.Solid_Box(x,y+100,x+T,y+200,c1)
    LCD.Solid_Box(x,y,x+T,y+100,0)
    LCD.Solid_Box(x+T,y+100-T,x+100-T,y+100,c1)
  if(N == 3):
      :
```

not the most most efficient code, but works

Display(Speed)

- Bottom-up programming
- Level 2

Pass the speed

- 00.0 to 99.9

Pull out the digits

- 10s, 1s, 0.1's

Display each digit

Add a decimal point

```
def Display(Speed):
    X = int(Speed*10)
    A0 = X % 10
    X = X // 10
    A1 = X % 10
    X = X // 10
    A2 = X % 10
    Big_Display(A2, 50, 50)
    Big_Display(A1, 170, 50)
    Big_Display(A0, 300, 50)
    LCD.Solid_Box(280,235,295,250,0xFFFF)
```

Main Loop

Read the GPS sensor until reading is valid

- Error_Flag isn't set

Checks button GP14

- Toggle recording on / off

Calculate & Display speed

- mph
-

```
while(Button15.value() == 1):
  Error_Flag = 1
  while(Error_Flag == 1):
    Error_Flag = 0
    [t, x, y, v] = GPS_Read(0)
  if(Button14.value() == 0):
    Record_Flag = not Record_Flag
    if(Record_Flag):
      Beep()
      f = open(FileName, "a")
      print('Recording')
      LCD.Text('Recording',5,5,Pink,Navy)
    else:
      Beep()
      sleep(0.1)
      Beep()
      f.close()
      print('File Closed')
      LCD.Text('         ',5,5,Pink,Navy)
  while(Button14.value() == 0):
    pass

  Display(v*1.15078)

  if(Record_Flag):
    msg = str('{:9.4f}'.format(v*1.15077) + ' ' )
    f.write(msg + '\n')
```

Speedometer Accuracy

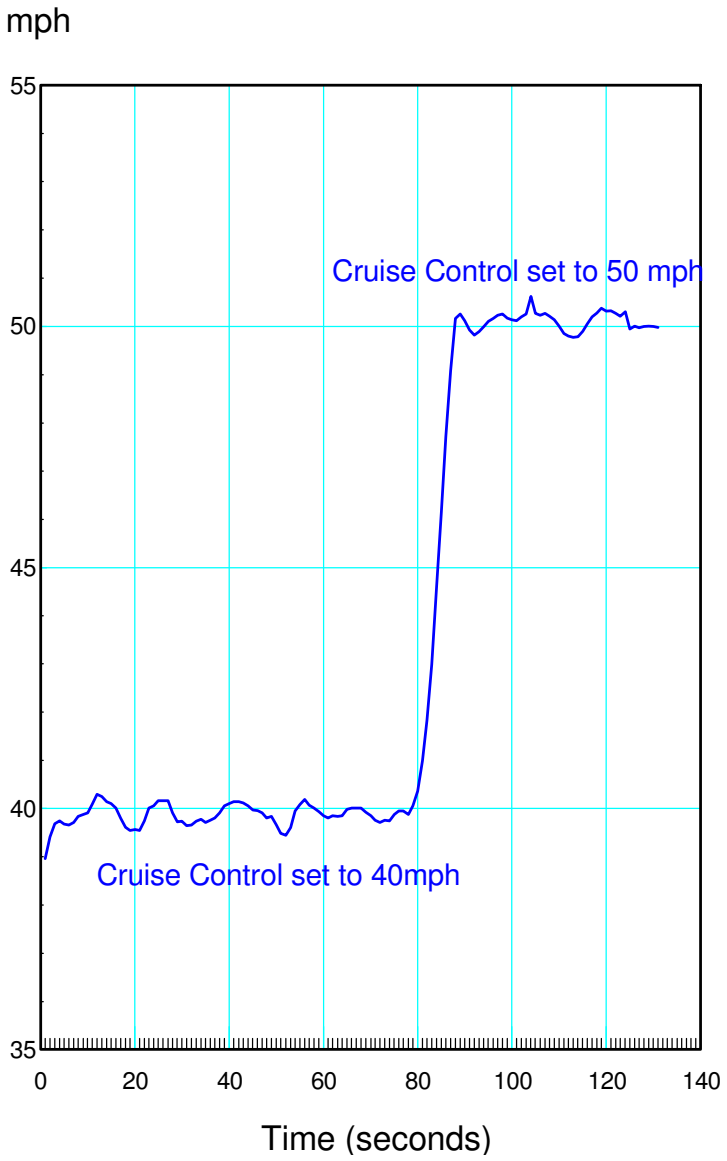
Need something to measure against

- Use car's cruise control
- Set to 40mph
- Set to 50mph

Result shows noise on readings

- Don't know cause
- Could be GPS sensor
- Could be actual speed is varying

	mean (mph)	st dev (mph)
40 mph	39.8913	0.1977
50 mph	50.0601	0.1954



Summary

GPS sensors are fairly inexpensive costing as little as $5 each from Amazon. With them you can determine where you are to within about 5 meters and your speed to within about 0.3 mph. Presumably, more expensive GPS sensors will work even better.

GPS sensors communicate with the Pi-Pico using SCI protocol. With some coding, the GPS messages can be pulled out and the fields can be read fairly easily. What you do with this is up to you and your creativity.

References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- https://learn.pimoroni.com/article/getting-started-with-pico
- https://www.w3schools.com/python/default.asp
- https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
- https://docs.micropython.org/en/latest/library/index.html
- https://www.fredscave.com/02-about.html

Pi-Pico Breadboard Kit

- https://wiki.52pi.com/index.php?title=EP-0172

Other

- https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/
- https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/
- https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/