**Final delivery**

**Sprint-1:**

#include <LiquidCrystal.h>

#include <Servo.h>

void UpDown();

void LeftRight();

Servo servo1;

Servo servo2;

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {

  lcd.begin(16,2);

  lcd.print("servo1 ");

  lcd.setCursor(0,1);

  lcd.print("servo2 ");

  servo1.attach(9);

  servo2.attach(10);

  servo1.write(90);

  servo2.write(90);

}

void loop(){

 int sensorTop = analogRead(A0);

 int sensorBottom = analogRead(A1);

 int sensorLeft = analogRead(A3);

 int sensorRight = analogRead(A4);

 int avgT=(sensorTop+sensorBottom)/2;

 int avgB=(sensorLeft+sensorRight)/2;

 int avgL=(sensorTop+sensorLeft)/2;

 int avgR=(sensorBottom+sensorRight)/2;

  if (avgT > avgB)

  {

    UpDown(sensorTop, sensorBottom);

  }

  if(avgT < avgB)

  {

    UpDown(sensorTop, sensorBottom);

  }

  if(avgL > avgR)

  {

   LeftRight(sensorLeft, sensorRight);

    }

  if(avgL < avgR)

  {

   LeftRight(sensorLeft, sensorRight);

    }

 delay(10);

}

void UpDown(int avgT, int avgB){

  int pos1= servo1.read();

  if(avgT < avgB){

     pos1 = --pos1;

    }

    else

     {

      pos1 = ++pos1;

     }

  servo1.write(pos1);

   lcd.setCursor(12,0);

 lcd.print(pos1);

 }

void LeftRight(int avgL, int avgR){

  int pos2= servo2.read();

  if(avgL < avgR)

   {

    pos2 = --pos2;

   }

   else

    {

     pos2 = pos2 + 1;

    }

  servo2.write(pos2);

  lcd.setCursor(12,1);

 lcd.print(pos2);

}

|  |  |
| --- | --- |
|  |  |

**Link:**[**https://www.tinkercad.com/things/5gsike0MSh8-magnificent-hango-snaget/editel?tenant=circuits**](https://www.tinkercad.com/things/5gsike0MSh8-magnificent-hango-snaget/editel?tenant=circuits)

**Sprint-2**

Diagram.json:

{

  "version": 1,

  "author": "Uri Shaked",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-arduino-mega", "id": "mega", "top": -1.43, "left": -48.12, "attrs": {} },

    { "type": "chip-gps-fake", "id": "chip1", "top": -75.78, "left": 196.8, "attrs": {} }

  ],

  "connections": [

    [ "chip1:GND", "mega:GND.2", "black", [ "v0", "h49.81", "v259.2", "h-124.8" ] ],

    [ "chip1:VCC", "mega:5V", "red", [ "h59.41", "v278.4", "h-134.4" ] ],

    [ "mega:19", "chip1:TX", "yellow", [ "v-45.97", "h-78.38", "v-19.2" ] ],

    [ "mega:18", "chip1:RX", "orange", [ "v-36.37", "h-77.98", "v-38.4" ] ]

  ]

}

GPS-FAKECHIP.H:

#include <stdio.h>

#include <stdlib.h>

#include "wokwi-api.h"

DEFINE\_PIN(RX);

DEFINE\_PIN(TX);

DEFINE\_PIN(VCC);

DEFINE\_PIN(GND);

#define LEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))

#define SECOND 1000000 /\* micros \*/

const char gps\_tx\_data[][80] = { // GPRMC & GPGGA (Hypothetical Data)

  "$GPGGA,172914.049,2327.985,S,05150.410,W,1,12,1.0,0.0,M,0.0,M,,\*60\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172914.049,A,2327.985,S,05150.410,W,009.7,025.9,060622,000.0,W\*74\r\n",

  "$GPGGA,172915.049,2327.982,S,05150.409,W,1,12,1.0,0.0,M,0.0,M,,\*6E\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172915.049,A,2327.982,S,05150.409,W,009.7,025.9,060622,000.0,W\*7A\r\n",

  "$GPGGA,172916.049,2327.980,S,05150.408,W,1,12,1.0,0.0,M,0.0,M,,\*6E\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172916.049,A,2327.980,S,05150.408,W,009.7,025.9,060622,000.0,W\*7A\r\n",

  "$GPGGA,172917.049,2327.977,S,05150.406,W,1,12,1.0,0.0,M,0.0,M,,\*69\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172917.049,A,2327.977,S,05150.406,W,009.7,025.9,060622,000.0,W\*7D\r\n",

  "$GPGGA,172918.049,2327.975,S,05150.405,W,1,12,1.0,0.0,M,0.0,M,,\*67\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172918.049,A,2327.975,S,05150.405,W,009.7,025.9,060622,000.0,W\*73\r\n",

  "$GPGGA,172919.049,2327.973,S,05150.404,W,1,12,1.0,0.0,M,0.0,M,,\*61\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172919.049,A,2327.973,S,05150.404,W,009.7,025.9,060622,000.0,W\*75\r\n",

  "$GPGGA,172920.049,2327.970,S,05150.403,W,1,12,1.0,0.0,M,0.0,M,,\*6F\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172920.049,A,2327.970,S,05150.403,W,009.7,025.9,060622,000.0,W\*7B\r\n",

  "$GPGGA,172921.049,2327.968,S,05150.402,W,1,12,1.0,0.0,M,0.0,M,,\*66\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172921.049,A,2327.968,S,05150.402,W,009.7,025.9,060622,000.0,W\*72\r\n",

  "$GPGGA,172922.049,2327.965,S,05150.401,W,1,12,1.0,0.0,M,0.0,M,,\*6B\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172922.049,A,2327.965,S,05150.401,W,009.7,025.9,060622,000.0,W\*7F\r\n",

  "$GPGGA,172923.049,2327.963,S,05150.399,W,1,12,1.0,0.0,M,0.0,M,,\*6A\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172923.049,A,2327.963,S,05150.399,W,009.7,025.9,060622,000.0,W\*7E\r\n",

  "$GPGGA,172924.049,2327.960,S,05150.398,W,1,12,1.0,0.0,M,0.0,M,,\*6F\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172924.049,A,2327.960,S,05150.398,W,009.7,294.1,060622,000.0,W\*7B\r\n",

  "$GPGGA,172925.049,2327.959,S,05150.401,W,1,12,1.0,0.0,M,0.0,M,,\*63\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172925.049,A,2327.959,S,05150.401,W,009.7,294.1,060622,000.0,W\*77\r\n",

  "$GPGGA,172926.049,2327.958,S,05150.403,W,1,12,1.0,0.0,M,0.0,M,,\*63\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172926.049,A,2327.958,S,05150.403,W,009.7,294.1,060622,000.0,W\*77\r\n",

  "$GPGGA,172927.049,2327.957,S,05150.406,W,1,12,1.0,0.0,M,0.0,M,,\*68\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172927.049,A,2327.957,S,05150.406,W,009.7,205.5,060622,000.0,W\*70\r\n",

  "$GPGGA,172928.049,2327.959,S,05150.407,W,1,12,1.0,0.0,M,0.0,M,,\*68\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172928.049,A,2327.959,S,05150.407,W,009.7,205.5,060622,000.0,W\*70\r\n",

  "$GPGGA,172929.049,2327.962,S,05150.408,W,1,12,1.0,0.0,M,0.0,M,,\*6E\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172929.049,A,2327.962,S,05150.408,W,009.7,205.5,060622,000.0,W\*76\r\n",

  "$GPGGA,172930.049,2327.964,S,05150.410,W,1,12,1.0,0.0,M,0.0,M,,\*69\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172930.049,A,2327.964,S,05150.410,W,009.7,205.5,060622,000.0,W\*71\r\n",

  "$GPGGA,172931.049,2327.967,S,05150.411,W,1,12,1.0,0.0,M,0.0,M,,\*6A\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172931.049,A,2327.967,S,05150.411,W,009.7,205.5,060622,000.0,W\*72\r\n",

  "$GPGGA,172932.049,2327.969,S,05150.412,W,1,12,1.0,0.0,M,0.0,M,,\*64\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172932.049,A,2327.969,S,05150.412,W,009.7,205.5,060622,000.0,W\*7C\r\n",

  "$GPGGA,172933.049,2327.971,S,05150.413,W,1,12,1.0,0.0,M,0.0,M,,\*6D\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172933.049,A,2327.971,S,05150.413,W,009.7,205.5,060622,000.0,W\*75\r\n",

  "$GPGGA,172934.049,2327.974,S,05150.414,W,1,12,1.0,0.0,M,0.0,M,,\*68\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172934.049,A,2327.974,S,05150.414,W,009.7,205.5,060622,000.0,W\*70\r\n",

  "$GPGGA,172935.049,2327.976,S,05150.415,W,1,12,1.0,0.0,M,0.0,M,,\*6A\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172935.049,A,2327.976,S,05150.415,W,009.7,205.5,060622,000.0,W\*72\r\n",

  "$GPGGA,172936.049,2327.979,S,05150.417,W,1,12,1.0,0.0,M,0.0,M,,\*64\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172936.049,A,2327.979,S,05150.417,W,009.7,205.5,060622,000.0,W\*7C\r\n",

  "$GPGGA,172937.049,2327.981,S,05150.418,W,1,12,1.0,0.0,M,0.0,M,,\*6D\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172937.049,A,2327.981,S,05150.418,W,009.7,117.1,060622,000.0,W\*71\r\n",

  "$GPGGA,172938.049,2327.983,S,05150.415,W,1,12,1.0,0.0,M,0.0,M,,\*6D\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172938.049,A,2327.983,S,05150.415,W,009.7,117.1,060622,000.0,W\*71\r\n",

  "$GPGGA,172939.049,2327.984,S,05150.413,W,1,12,1.0,0.0,M,0.0,M,,\*6D\r\n",

  "$GPGSA,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0\*30\r\n",

  "$GPRMC,172939.049,A,2327.984,S,05150.413,W,009.7,117.1,060622,000.0,W\*71\r\n",

};

typedef struct {

  uart\_dev\_t uart0;

  uint32\_t gps\_tx\_index;

  uint32\_t tx\_timer;

} chip\_state\_t;

static void on\_uart\_rx\_data(void \*ctx, uint8\_t byte);

static void on\_uart\_write\_done(void \*ctx);

void EXPORT(chip\_timer\_event)(chip\_state\_t \*chip, uint32\_t timer\_id) {

  if (timer\_id == chip->tx\_timer) {

    const char \*message = gps\_tx\_data[chip->gps\_tx\_index++];

    uart\_write(chip->uart0, message, strlen(message));

    if (chip->gps\_tx\_index >= LEN(gps\_tx\_data)) {

      chip->gps\_tx\_index = 0;

    }

  }

}

void\* chip\_init(void) {

  setvbuf(stdout, NULL, \_IOLBF, 1024);

  chip\_state\_t \*chip = malloc(sizeof(chip\_state\_t));

  chip->gps\_tx\_index = 0;

  chip->tx\_timer = timer\_init();

  timer\_start(chip->tx\_timer, SECOND, true);

  const uart\_config\_t uart\_config = {

    .tx = pin\_init("TX", INPUT\_PULLUP),

    .rx = pin\_init("RX", INPUT),

    .baud\_rate = 4800,

  };

  chip->uart0 = uart\_init(chip, &uart\_config);

  return chip;

}

GPS FAKE CHIP.JSON:

{

  "name": "GPS Fake",

  "author": "Anderson Costa",

  "pins": [

    "RX",

    "TX",

    "GND",

    "VCC"

  ]

}

NMEA.H:

#ifdef NMEA\_h

#define NMEA\_h

#include "Arduino.h"

#define GPRMC       1

#define MTR         1.0

#define KM          0.001

#define MI          0.00062137112

#define NM          0.00053995680

#define PARSEC      0.000000000000

#define MPS         0.51444444

#define KMPH        1.852

#define KTS         1.0

#define LIGHTSPEED  0.000000001716

class NMEA

{

  public:

    NMEA(int connect);

    int decode(char c);

    float gprmc\_utc();

    char gprmc\_status();

    float gprmc\_latitude();

    float gprmc\_longitude();

    float gprmc\_speed(float unit

    float gprmc\_course();

    float gprmc\_distance\_to(float latitude, float longitude, float unit;

    float gprmc\_course\_to(float latitude, float longitude);

    char\* sentence();

    int terms();

    char\* term(int t);

    float term\_decimal(int t);

    int libversion();

  private:

    int \_gprmc\_only;

    float \_gprmc\_utc;

    char \_gprmc\_status;

    float \_gprmc\_lat;

    float \_gprmc\_long;

    float \_gprmc\_speed;

    float \_gprmc\_angle;

    char  f\_sentence[100];

    char\* f\_term[30];

    int f\_terms;

    int \_terms;

    char \_sentence[100];

    char\* \_term[30];

    int n;

    int \_gprmc\_tag;

    int \_state;

    int \_parity;

    int \_nt;

    float \_degs;

    float distance\_between(float lat1, float long1, float lat2, float long2, float units\_per\_meter);

    float initial\_course(float lat1, float long1, float lat2, float long2);

    int \_dehex(char a);

    float \_decimal(char\* s);

};

#endif

NMEA.CPP:

#include "Arduino.h"

#include "NMEA.h"

#define \_GPRMC\_TERM   "$GPRMC,"

#define \_GNRMC\_TERM   "$GNRMC,"

#define \_LIB\_VERSION  1

NMEA::NMEA(int connect)

{

  \_gprmc\_only = connect;

  \_gprmc\_utc = 0.0;

  \_gprmc\_status = 'V';

  \_gprmc\_lat = 0.0;

  \_gprmc\_long = 0.0;

  \_gprmc\_speed = 0.0;

  \_gprmc\_angle = 0.0;

  \_terms = 0;

  n = 0;

  \_state = 0;

  \_parity = 0;

  \_nt = 0;

  f\_sentence[0] = 0;

  f\_terms = 0;

  for (int t = 0; t < 30; t++) {

    \_term[t] = (char\*) malloc (15 \* sizeof(char));

    f\_term[t] = (char\*) malloc (15 \* sizeof(char));

    (f\_term[t])[0] = 0;

  }

}

int NMEA::decode(char c) {

  if ((n >= 100) || (\_terms >= 30) || (\_nt >= 15)) {

    \_state = 0;

  }

  if ((c == 0x0A) || (c == 0x0D)) {

    \_state = 0;

  }

  if (c == '$') {

    \_gprmc\_tag = 0;

    \_parity = 0;

    \_terms = 0;

    \_nt = 0;

    \_sentence[0] = c;

    n = 1;

    \_state = 1;

    return 0;

  }

  switch (\_state) {

    case 0;

      break;

    case 1;

      if (n < 7) {

        if ((c == \_GNRMC\_TERM[n]) || (c == \_GPRMC\_TERM[n])) {

          \_gprmc\_tag++;

        }

      }

      \_sentence[n++] = c;

      switch (c) {

        case ',':

          (\_term[\_terms++])[\_nt] = 0;

          \_nt = 0;

          \_parity = \_parity ^ c;

          break;

        case '\*';

          (\_term[\_terms++])[\_nt] = 0;

          \_nt = 0;

          \_state++;

          break;

        default:

          (\_term[\_terms])[\_nt++] = c;

          \_parity = \_parity ^ c;

          break;

      }

      break;

    case 2:

      \_sentence[n++] = c;

      (\_term[\_terms])[\_nt++] = c;

      \_parity = \_parity - (16 \* \_dehex(c));

      \_state++;

      break;

    case 3:

      \_sentence[n++] = c;

      \_sentence[n++] = 0;

      (\_term[\_terms])[\_nt++] = c;

      (\_term[\_terms++])[\_nt] = 0;

      \_state = 0;

      \_parity = \_parity - \_dehex(c);

      if (\_parity == 0) {

        if ((!\_gprmc\_only) || (\_gprmc\_tag == 6)) {

          while ((--n) >= 0) {

            f\_sentence[n] = \_sentence[n];

          }

          for (f\_terms = 0; f\_terms < \_terms; f\_terms++) {

            \_nt = 0;

            while ((\_term[f\_terms])[\_nt]) {

              (f\_term[f\_terms])[\_nt] = (\_term[f\_terms])[\_nt];

              \_nt++;

            }

            (f\_term[f\_terms])[\_nt] = 0;

          }

          if (\_gprmc\_tag == 6){

            \_gprmc\_utc = \_decimal(\_term[1]);

            \_gprmc\_status = (\_term[2])[0];

            \_gprmc\_lat = \_decimal(\_term[3]) / 100.0;

            \_degs = floor(\_gprmc\_lat);

            \_gprmc\_lat = (100.0 \* (\_gprmc\_lat - \_degs)) / 60.0;

            \_gprmc\_lat += \_degs;

            if ((\_term[4])[0] == 'S') {

              \_gprmc\_lat = 0.0 - \_gprmc\_lat;

            }

            \_gprmc\_long = \_decimal(\_term[5]) / 100.0;

            \_degs = floor(\_gprmc\_long);

            \_gprmc\_long = (100.0 \* (\_gprmc\_long - \_degs)) / 60.0;

            \_gprmc\_long += \_deg;

            if ((\_term[6])[0] == 'W') {

              \_gprmc\_long = 0.0 - \_gprmc\_long;

            }

            \_gprmc\_speed = \_decimal(\_term[7]);

            \_gprmc\_angle = \_decimal(\_term[8]);

          }

          return 1;

        }

      }

      break;

    default:

      \_state = 0;

      break;

  }

  return 0;

}

float NMEA::gprmc\_utc(){

  return \_gprmc\_utc;

}

char NMEA::gprmc\_status() {

  return \_gprmc\_status;

}

float NMEA::gprmc\_latitude() {

  return \_gprmc\_lat;

}

float NMEA::gprmc\_longitude() {

    return \_gprmc\_long;

}

float NMEA::gprmc\_speed(float unit) {

  return (\_gprmc\_speed \* unit);

}

float NMEA::gprmc\_course() {

  return \_gprmc\_angle;

}

float NMEA::gprmc\_distance\_to(float latitude, float longitude, float unit) {

  return distance\_between( \_gprmc\_lat, \_gprmc\_long, latitude, longitude, unit);

}

float NMEA::gprmc\_course\_to(float latitude, float longitude) {

  return initial\_course( \_gprmc\_lat, \_gprmc\_long, latitude, longitude);

}

char\* NMEA::sentence() {

  return f\_sentence;

}

int NMEA::terms() {

  return f\_terms;

}

char\* NMEA::term(int t) {

  return f\_term[t];

}

float NMEA::term\_decimal(int t) {

  return \_decimal(f\_term[t]);

}

int NMEA::libversion() {

  return \_LIB\_VERSION;

}

float NMEA::distance\_between (float lat1, float long1, float lat2, float long2, float units\_per\_meter)

{  float delta = radians(long1 - long2);

  float sdlong = sin(delta);

  float cdlong = cos(delta);

  lat1 = radians(lat1);

  lat2 = radians(lat2);

  float slat1 = sin(lat1);

  float clat1 = cos(lat1);

  float slat2 = sin(lat2);

  float clat2 = cos(lat2);

  delta = (clat1 \* slat2) - (slat1 \* clat2 \* cdlong);

  delta = sq(delta);

  delta += sq(clat2 \* sdlong);

  delta = sqrt(delta);

  float denom = (slat1 \* slat2) + (clat1 \* clat2 \* cdlong);

  delta = atan2(delta, denom);

  return delta \* 6372795 \* units\_per\_meter;

}

float NMEA::initial\_course (float lat1, float long1, float lat2, float long2) {

  float dlon = radians(long2 - long1);

  lat1 = radians(lat1);

  lat2 = radians(lat2);

  float a1 = sin(dlon) \* cos(lat2);

  float a2 = sin(lat1) \* cos(lat2) \* cos(dlon);

  a2 = cos(lat1) \* sin(lat2) - a2;

  a2 = atan2(a1, a2);

  if (a2 < 0.0) {

    a2 += TWO\_PI;

  }

  return degrees(a2);

}

int NMEA::\_dehex(char a) {

  if (int(a) >= 65) {

    return int(a) - 55;

  }

  else {

    return int(a) - 48;

  }

}

float NMEA::\_decimal(char\* s) {

  long  rl = 0;

  float rr = 0.0;

  float rb = 0.1;

  boolean dec = false;

  int i = 0;

  if ((s[i] == '-') || (s[i] == '+')) {

    i++;

  }

  while (s[i] != 0) {

    if (s[i] == '.') {

      dec = true;

    }

    else {

      if (!dec) {

        rl = (10 \* rl) + (s[i] - 48);

      }

      else {

        rr += rb \* (float)(s[i] - 48);

        rb /= 10.0;

      }

    }

    i++;

  }

  rr += (float)rl;

  if (s[0] == '-') {

    rr = 0.0 - rr;

  }

  return rr;

}

GPS.CHIPS.EXAMPLE.IN:

#include "NMEA.h"

#define LEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))

union {

  char bytes[4];

  float valor;

} velocidadeGPS;

float latitude;

float longitude;

NMEA gps(GPRMC);

void setup() {

**Serial**.begin(9600);

  Serial1.begin(4800);

**Serial**.println("Data received from GPS Fake:");

}

void loop() {

  while (Serial1.available()) {

    char serialData = Serial1.read();

**Serial**.print(serialData);

    if (gps.decode(serialData)) {

      if (gps.gprmc\_status() == 'A') {

        velocidadeGPS.valor = gps.gprmc\_speed(KMPH);

      } else {

        velocidadeGPS.valor = 0;

      }

      latitude = gps.gprmc\_latitude();

      longitude = gps.gprmc\_longitude();

**Serial**.println();

**Serial**.println();

**Serial**.print(" Latitude: ");

**Serial**.println(latitude, 8);

**Serial**.print("Longitude: ");

**Serial**.println(longitude, 8);

**Serial**.print("    Speed: ");

**Serial**.print(velocidadeGPS.valor);

**Serial**.println(" Km/h");

      convertCoordinatesToCartesian(latitude, longitude);

    }

  }

}

void convertCoordinatesToCartesian(float latitude, float longitude) {

  float latRadius = latitude \* (PI) / 180;

  float lonRadius = longitude \* (PI) / 180;

  int earthRadius = 6371; // Radius in km

  float posX = earthRadius \* cos(latRadius) \* cos(lonRadius);

  float posY = earthRadius \* cos(latRadius) \* sin(lonRadius);

**Serial**.print("        X: ");

**Serial**.println(posX);

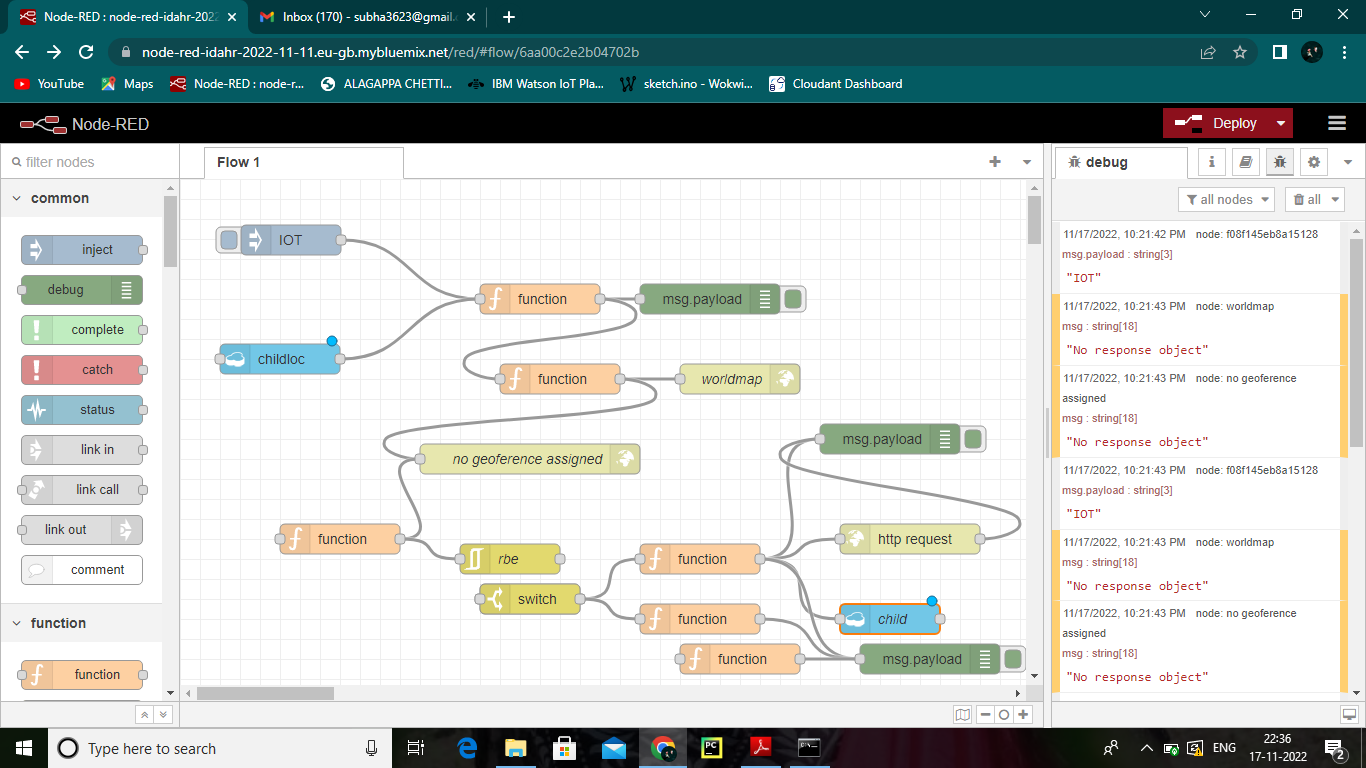
**Serial**.print("        Y: ");

**Serial**.println(posY);

}

LINK:<https://wokwi.com/projects/348673807535309395>

**Sprint-3**

****

**Sprint-4**

**Web application:**

|  |  |  |
| --- | --- | --- |
| |  | | --- | |  | |  |

<https://childsafetymonitoringandnotification.000webhostapp.com/step1.html>