Sprint-2

Date	18.11.2022
TeamID	PNT2022TMID50079
ProjectName	IOT Based safety gadget for child safety monitoring and notification

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);

#defineLEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))
#define SECOND 1000000/* micros */
```

```
constchargps_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",
};

typedefstruct {
  uart_dev_t uart0;
  uint32_t gps_tx_index;
  uint32_t tx_timer;
} chip_state_t;

staticvoidon_uart_rx_data(void *ctx, uint8_t byte);
staticvoidon_uart_write_done(void *ctx);
```

```
voidEXPORT(chip_timer_event)(chip_state_t *chip, uint32_t timer_id) {
  if (timer_id == chip->tx_timer) {
    constchar *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);

  constuart_config_tuart_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:

```
#ifdefNMEA_h
#defineNMEA_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
#defineLIGHTSPEED  0.000000001716
class NMEA
{
  public:
    NMEA(int connect);
    intdecode(char c);
    floatgprmc_utc();
    chargprmc_status();
    floatgprmc_latitude();
    floatgprmc_longitude();
    floatgprmc_speed(float unit
    floatgprmc_course();
    floatgprmc_distance_to(float latitude, float longitude, float unit;
    floatgprmc_course_to(float latitude, float longitude);
    char* sentence();
    intterms();
    char* term(int t);
    floatterm_decimal(int t);
    intlibversion();
  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];
    intf_terms;
    int _terms;
    char _sentence[100];
    char* _term[30];
    int n;
```

```
    int _gprmc_tag;
    int _state;
    int _parity;
    int _nt;
    float _degs;

    floatdistance_between(float lat1, float long1, float lat2, float long2,
floatunits_per_meter);
    floatinitial_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;
  }
}
```

```
intNMEA::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;
    return0;
  }
  switch (_state) {
    case0;
      break;
    case1;
      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;
```

```
case2:
  _sentence[n++] = c;
  (_term[_terms])[_nt++] = c;
  _parity = _parity - (16 * _dehex(c));
  _state++;
  break;
case3:
  _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]);
          }

          return1;
        }
      }
      break;
    default:
      _state = 0;
      break;
  }
  return0;
}

floatNMEA::gprmc_utc(){
  return _gprmc_utc;
}

charNMEA::gprmc_status() {
  return _gprmc_status;
}

floatNMEA::gprmc_latitude() {
  return _gprmc_lat;
}

floatNMEA::gprmc_longitude() {
    return _gprmc_long;
}

floatNMEA::gprmc_speed(float unit) {
  return (_gprmc_speed * unit);
}

floatNMEA::gprmc_course() {
  return _gprmc_angle;
}

floatNMEA::gprmc_distance_to(float latitude, float longitude, float unit) {
  returndistance_between( _gprmc_lat, _gprmc_long, latitude, longitude, unit);
}

floatNMEA::gprmc_course_to(float latitude, float longitude) {
  returninitial_course( _gprmc_lat, _gprmc_long, latitude, longitude);
}
```

```
char* NMEA::sentence() {
  returnf_sentence;
}

intNMEA::terms() {
  returnf_terms;
}

char* NMEA::term(int t) {
  returnf_term[t];
}

floatNMEA::term_decimal(int t) {
  return _decimal(f_term[t]);
}

intNMEA::libversion() {
  return _LIB_VERSION;
}

floatNMEA::distance_between (float lat1, float long1, float lat2, float long2,
floatunits_per_meter)
{  float delta = radians(long1 - long2);
  floatsdlong = sin(delta);
  floatcdlong = 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);

  floatdenom = (slat1 * slat2) + (clat1 * clat2 * cdlong);

  delta = atan2(delta, denom);

  return delta * 6372795 * units_per_meter;
}
```

```
floatNMEA::initial_course (float lat1, float long1, float lat2, float long2) {
  floatdlon = 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;
  }
  returndegrees(a2);
}

intNMEA::_dehex(char a) {
  if (int(a) >= 65) {
    returnint(a) - 55;
  }
  else {
    returnint(a) - 48;
  }
}

floatNMEA::_decimal(char* s) {
  long  rl = 0;
  floatrr = 0.0;
  floatrb = 0.1;
  boolean dec = false;
  inti = 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;
```

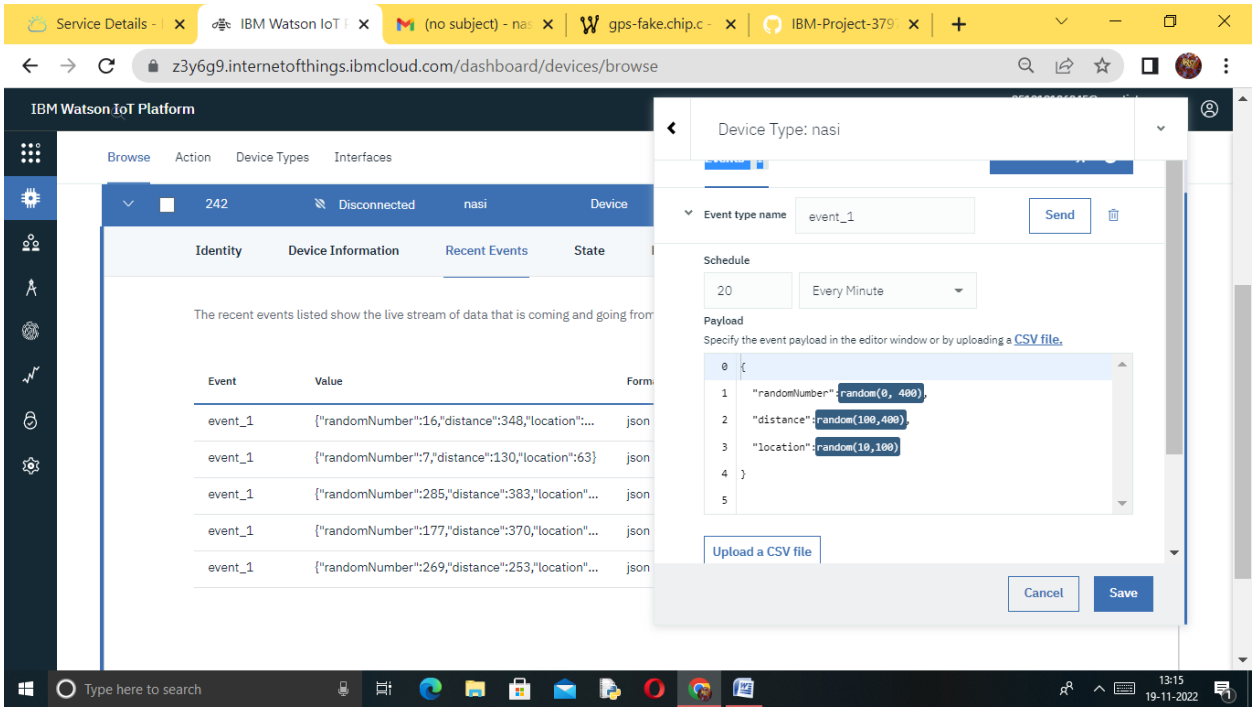
```
      }
    }
    i++;
  }
  rr += (float)rl;
  if (s[0] == '-') {
    rr = 0.0 - rr;
  }
  returnrr;
}
```

GPS.CHIPS.EXAMPLE.IN:

```
#include"NMEA.h"

#defineLEN(arr) ((int)(sizeof(arr) / sizeof(arr)[0]))

union {
  charbytes[4];
  float valor;
} velocidadeGPS;

float latitude;
float longitude;
NMEA gps(GPRMC);


voidsetup() {
  Serial.begin(9600);
  Serial1.begin(4800);
  Serial.println("Data received from GPS Fake:");
}

voidloop() {
  while (Serial1.available()) {
    charserialData = 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);
    }
  }
}

voidconvertCoordinatesToCartesian(float latitude, float longitude) {
  floatlatRadius = latitude * (PI) / 180;
  floatlonRadius = longitude * (PI) / 180;

  intearthRadius = 6371; // Radius in km

  floatposX = earthRadius * cos(latRadius) * cos(lonRadius);
  floatposY = earthRadius * cos(latRadius) * sin(lonRadius);

  Serial.print("        X: ");
  Serial.println(posX);

  Serial.print("        Y: ");
  Serial.println(posY);
}
```

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


Service Details -
IBM Watson IoT
(no subject) - nas
gps-fake.chip.c -
IBM-Project-379
z3y6g9.internetofthings.ibmcloud.com/dashboard/devices/browse
IBM Watson IoT Platform
Browse
Action
Device Types
Interfaces
242
Disconnected
nasi
Device
Identity
Device Information
Recent Events
State
The recent events listed show the live stream of data that is coming and going from
Event
Value
event_1
{"randomNumber":16,"distance":348,"location":...
{"randomNumber":7,"distance":130,"location":63}
{"randomNumber":285,"distance":383,"location"...
{"randomNumber":177,"distance":370,"location"...
{"randomNumber":269,"distance":253,"location"...
json
Device Type: nasi
Event type name
event_1
Send
Schedule
20
Every Minute
Payload
Specify the event payload in the editor window or by uploading a CSV file.
"randomNumber":random(0, 400),
"distance":random(100,400),
"location":random(10,100)
Upload a CSV file
Cancel
Save
Type here to search
13:15
19-11-2022