

## 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 & GPGLL (Hypothetical Data)
  "$GPGLL,172914.049,2327.985,S,05150.410,W,1,12,1.0,0.0,M,0.0,M,,*60\r\n",
  "$GPGLL,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$GPRMC,172914.049,A,2327.985,S,05150.410,W,009.7,025.9,060622,000.0,W*74\r\n",
  "$GPGLL,172915.049,2327.982,S,05150.409,W,1,12,1.0,0.0,M,0.0,M,,*6E\r\n",
  "$GPGLL,A,3,01,02,03,04,05,06,07,08,09,10,11,12,1.0,1.0,1.0*30\r\n",
  "$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+1]][_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 r1 = 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) {
                r1 = (10 * r1) + (s[i] - 48);
            }
            else {
                rr += rb * (float)(s[i] - 48);
                rb /= 10.0;
            }
        }
        i++;
    }
    rr += (float)r1;
    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>

The screenshot displays the Wokwi web IDE interface. On the left, the code editor shows the contents of 'gps-fake.chip.c', which includes headers for stdio.h and stdlib.h, and a large array of simulated GPS data points. On the right, a 3D simulation of an Arduino MEGA board is shown, with a 'GPS Fake Breakout' module connected to its pins. The top of the browser window shows several open tabs, including 'wokwi.com/projects/348673807535309395', 'sketchino - Wokwi Arduino', and 'Python and GPS Tracking - S...'. The bottom of the browser window shows the Windows taskbar with various application icons.

The screenshot shows the IBM Watson IoT Platform dashboard. The main view is for a device named 'subha\_2', which is in a 'Connected' state. A modal window is open for configuring an event named 'event\_1'. The event is set to occur '20 x Every Minute'. The payload is configured as a JSON object with the following fields:

```

1 px.scatter_mapbox(gdf, lat="latitude", lon="longitude", color="ge
2 size_max=6, zoom=12, width=1200, height=800)
3 }
4

```

The modal also includes a 'Send' button, a 'Cancel' button, and a 'Save' button. A link 'What functions can I apply?' is visible at the bottom of the modal.

```

python1.py - C:/Users/Karuna/Desktop/python1.py (3.7.4)
File Edit Format Run Options Window Help

import json
import wiotp.sdk.device
import time

myConfig = {
    "identity": {
        "orgId": "bo04id",
        "typeId": "MyDevice",
        "deviceId": "1234"
    },
    "auth": {
        "token": "wY80cR6TsklnXT)u)8"
    }
}

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    name= "Smartbridge"
    #in area location

    #latitude= 17.4225176
    #longitude= 76.5458842

    #out area location

    latitude= 17.4219272
    longitude= 76.5488783
    myData={name: name, 'lat':latitude, 'lon':longitude}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Data published to IBM IOT platform: ",myData)
    time.sleep(5)

client.disconnect()

```

Activate Windows  
Go to Settings to activate Windows.

Ln: 2 Col: 23

