**CHILD TRACKER WITH EMERGENCY NOTIFIER**

**Abstract:**

The crime rate in the world has been increasing day by day, moreover safety of a child is in jeopardy. It is estimated that around 60000 kids are abducted every year and the count is increasing. Kidnapping, trafficking on one hand and the hectic life of parents makes it very difficult to personally look after the kids. To overcome the problem of personal monitoring of children, we have come up with an idea of GPS based tracker with text alerts when the child is not present in the specified region.

**Problem Statement:**

To solve the problem of difficulty in keeping a check on children while they are playing in open areas while letting the children have their freedom to play from constantly being mocked by parents and to monitor the whereabouts of children by working parents.

**Project Working Process:**

GPS module

GSM/GPRS module

Arduino

UNO

Cloud

Mobile

App

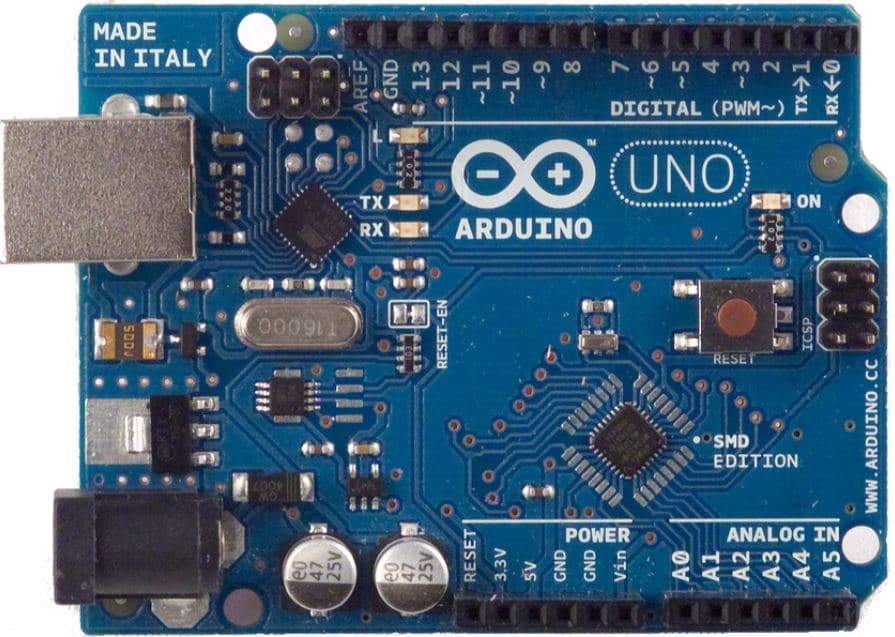
In this context, the solution we propose is an android application which can help the parents in tracking the presence of their children. This application works with the help of android mobile enabled with ‘SMS’ feature and ‘Global-Positioning System (GPS)’ on GSM network. This application works in two ways, first GPS based that is related to the Location services and second SMS based which is related to the Network services. In case when the Location based services not working then the application alternatively uses Network based services which can send and receive messages.

The GSM and GPS module are connected to Arduino Uno. The GSM module has a sim (data) through which we can receive message alerts and the GPS helps to track the location. The data is sent to the cloud from GSM using AT commands and http. The cloud is connected to mobile app and the data is displayed in the app installed in parents mobile. The app is integrated with maps and geo-fence. A geo-fence is a virtual geographic boundary defined by a parent using GPS that enables software to trigger response when a device enters or leaves a particular area. We can implement using node MCU and connecting it to GPS.

The GPS is connected to Arduino using VCC pin, GND pin, RX pin, TX pin. Connections: VCC to 3V3, GND to GND, RX to TX and TX to RX respectively. GSM is connected to Arduino using VCC, GND, RX, TX pins. Connections: VCC to 5V, GND to GND, pin 3 to TX, pin 4 to RX respectively. Tiny GPS++ library is included. Using IBM cloud (node red flow editor) the data is sent to mobile app developed using MIT app inventor.

**Components:**

**Arduino UNO:**

An Arduino is actually a microcontroller-based kit which can be either used directly by purchasing from the vendor or can be made at home using the components, owing to its open source hardware feature. It is basically used in communications and in controlling or operating many devices. It was founded by Massimo Banzi and David Cuartielles in 2005.

* Microcontroller: ATmega328
* Operating Voltage: 5V
* Input Voltage (recommended): 7-12V
* Input Voltage (limits): 6-20V
* Digital I/O Pins: 14 (of which 6 provide PWM output)
* Analog Input Pins: 6
* DC Current per I/O Pin: 40 mA
* DC Current for 3.3V Pin: 50 mA
* Flash Memory: 32 KB of which 0.5 KB used by bootloader
* SRAM: 2 KB (ATmega328)
* EEPROM: 1 KB (ATmega328)
* Clock Speed: 16 MHz

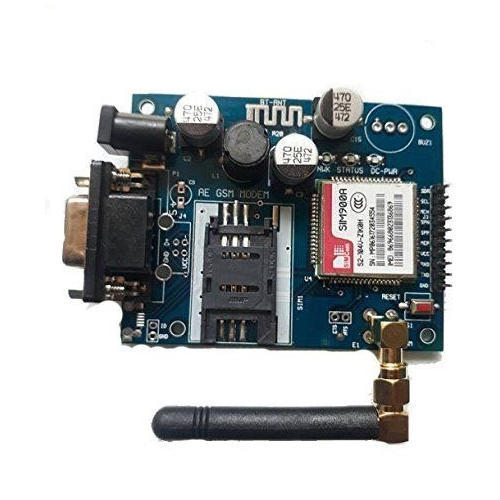
**GPS (Global Positioning System): UBlox NEO-6M**

The working/operation of Global positioning system is based on the ‘trilateration’ mathematical principle. The position is determined from the distance measurements to satellites. Global positioning system consists of satellite, control station and monitor station and receiver. The GPS receiver takes the information from the satellite and uses the method of triangulation to determine a user’s exact position.

* Frequency 1575.42MHz-L1 C/A code
* I/O Port UART interface Warm Start Time 38 sec
* GPS Channel 16 Channels Reacquisition 100ms
* Operating Voltage 3.0 V ~ 6.0 V Update Rate 1Hz
* Operating Temperature -40°C ~ +85°C
* External Antenna Current Range 2mA ~ 25mA
* Power Consumption 27mA I/O Connector 1.27mm
* Position Accuracy 3m CEP (50%), 7m EP (90%)
* Tracking Sensitivity -147dBm
* V Bat Voltage 1.0 V (MIN.), 2.0 V (MAX.)
* Dimensions 25mmX25mmX2.5mm

**GSM: SIM900A module**

GSM is a mobile communication modem. It stands for ‘Global System for Mobile’ communication (GSM). The idea of GSM was developed at Bell Laboratories in 1970. It is widely used mobile communication system in the world. GSM is an open and digital cellular technology used for transmitting mobile voice and data services operates at the 850MHz, 900MHz, 1800MHz and 1900MHz frequency bands.



* Built in RS232 to TTL and vice versa Logic Converter (MAX232)
* Configurable Baud Rate
* SMA (Sub Miniature version A) connector with GSM L Type Antenna
* Built in SIM (Subscriber Identity Module) Card holder
* Built in Network Status LED
* Inbuilt Powerful TCP / IP (Transfer Control Protocol / Internet Protocol) stack for
* Internet data transfer through GPRS (General Packet Radio Service)

**Node MCU: ESP8266**

NodeMCU Development Kit/Board consist of ESP8266 wifi chip. ESP8266 chip has GPIO pins, serial communication protocol, etc. features on it. **ESP8266** is a low-cost [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer [ESP8266 WiFi Module](http://www.electronicwings.com/sensors-modules/esp8266-wifi-module).

**FINAL CODE**:

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include<SoftwareSerial.h>0

#include<TinyGPS++.h>

const char\* host = "api.msg91.com";

const char\* ssid = "Apurva";

const char\* password = "apurva\*23\*";

TinyGPSPlus gps;

SoftwareSerial ss(4,5);

#define ORG "blm9z2"

#define DEVICE\_TYPE "Apurva"

#define DEVICE\_ID "001"

#define TOKEN "8333875659"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/evt/Data/fmt/json";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

int flag=0;

WiFiClient wifiClient;

PubSubClient client(server, 1883,wifiClient); //servername,portnumber

void setup() {

Serial.begin(9600);

Serial.println();

delay(1500);

ss.begin(9600);

Serial.print("Connecting to ");

Serial.print(ssid); //just to know to which wifi we are connected

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("WiFi connected, IP address: ");

Serial.println(WiFi.localIP());

}

void loop() {

Serial.print("connecting to ");

Serial.println(host);

double latitude=40.2436834;

double longitude=32.3479458;

Serial.print("Latitude:");

Serial.println(latitude,4);

Serial.print("Longitude:");

Serial.println(longitude,4);

smartDelay(1000);

if (millis() > 5000 && gps.charsProcessed() < 10)

Serial.println(F("No GPS data received: check wiring"));

if (isnan(latitude) || isnan(longitude))

{

Serial.println("Failed to read from GPS sensor!");

delay(1000);

return;

}

PublishData(latitude,longitude);

delay(100);

WiFiClient client;

String url = "https://api.msg91.com/api/sendhttp.php?mobiles=8919024936&authkey=280113AdefnZ9vhga5cfb6000&route=4&sender=TESTIN&message=Your child is not the specified area.Please check the app for the current location&country=91";

if(flag==0){

Serial.print("Requesting URL: ");

Serial.println(url);

const int httpPort = 80;

if (!client.connect(host, httpPort)) {

Serial.println("connection failed");

return;

}

client.print(String("GET ") + url + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

delay(10);

while(client.available()){

String line = client.readStringUntil('\r');

Serial.print(line);

}

flag++;

}

Serial.println();

Serial.println("closing connection");

}

static void smartDelay(unsigned long ms) // This custom version of delay() ensures that the gps object is being "fed".

{

unsigned long start = millis();

do

{

while (ss.available())

gps.encode(ss.read());

} while (millis() - start < ms);

}

void PublishData(float latitude, float longitude){

if (!!!client.connected()) {

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {//to connect client

Serial.print(".");

delay(500);

}

Serial.println();

}

String payload = "{\"d\":{\"latitude\":";

payload += latitude;

payload+="," "\"longitude\":";

payload += longitude;

payload += "}}";

Serial.print("Sending payload: ");

Serial.println(payload);

if (client.publish(topic, (char\*) payload.c\_str())) {

Serial.println("Publish ok");

} else {

Serial.println("Publish failed");

}

}