Volume 9, Issue 6, June – 2024
ISSN No:-2456-2165

https://doi.org/10.38124/ijisrt/IJISRT24JUN177

Temperature Call Alert to Prevent Child Death from Heatstroke in Smart Car using IoT

Srija Ghattamaneni[1]; Nesreen Alsbou[2]
University of Central Oklahoma

Abstract:- According to a study conducted by the National Health Service (NHS), 937 children died of heatstroke in the United States between 1998 and January 2023. Shockingly, out of these 937 deaths, 493 occurred due to parents forgetting to take their children out of cars. To prevent such tragic incidents from occurring, this paper proposes a simple system, "Temperature Call Alert". The system operates by utilizing multiple sensors to collect and analyze data regarding the temperature inside a car. This data is then sent to a cloud platform for storage and processing. If the system detects a higher-than-normal temperature inside the car, it automatically sends an alert call to the parent's mobile number with a message. This alert call can help the parent to take necessary actions to prevent their child's life from being lost due to heatstroke. To implement the temperature call alert system, we used an IFTTT webhook with an IoT cloud. This allows us to give a call alert to the authorized mobile number whenever the system detects a potentially dangerous situation. With the help of this system, we can reduce the risk of heatstroke deaths caused by parents forgetting their children in cars.

Keywords:- *Iot, Child Safety, Heatstroke, Arduino MKR1000, Sensors, Arduino Iot Cloud, IFTTT, Alert Message.*

I. INTRODUCTION

Heatstroke is a grove concern for children who are left in hot cars, as it can lead to death if not treated immediately . To address this issue, we propose an IoT system that uses a wifi- compatible device with multiple sensors to prevent heatstroke deaths in smart cars. Our research indicates that the majority of heatstroke cases occur during summer, with intermediate numbers in spring and fall, and the fewest cases in winter. Heat-related illness can occur within an average of 20 minutes, with an average heat rate of 40 °C/h. If left untreated, heatstroke could develop within 105 minutes, with an average heat rate of 4.8 °C/h. Without medical intervention, death could occur within 125 minutes. Therefore, sending a temperature alert to the driver's mobile number can help them take necessary action in a timely manner and prevent any loss of life.

Several papers have discussed the detection of human life in different situations. For instance, in one study [2], Author used multi sensor wearable for child safety, A wristband needs to be carried, based on the SMS and location they can trace. Another author [3] proposed an IoT system using Raspberry Pi with sensors. Another author [4] used Infrared Sensors to detect the heatstroke and send an SMS.

Our proposed system employs a microcontroller - Arduino MKR1000 - along with sensors such as vibration, motion, FSR, Temperature, and Actuators such as Buzzer, and an LED. The system collects the car's interior sensor data and analyzes it using Arduino IoT cloud technology. Additionally, we have used the IFTTT webhook to receive a real-time call alerts in case of temperature fluctuations. This approach provides an effective and cost-efficient means of sending temperature alerts to the driver's mobile device. Our proposed system aims to enhance the safety and well-being of children in smart cars and prevent potential tragedies.

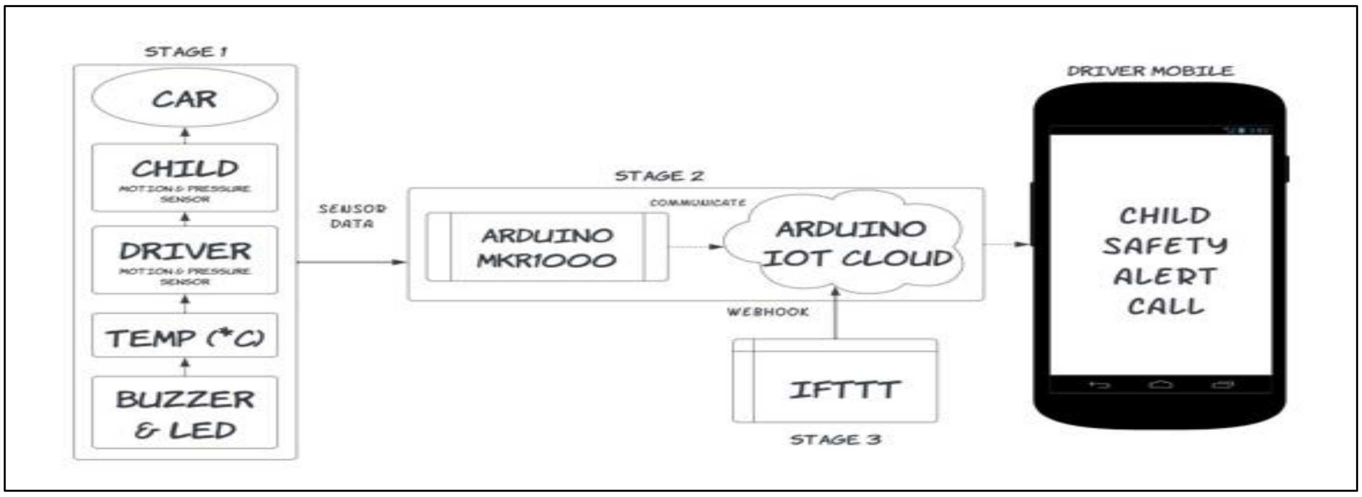


Fig 1: Proposed System Architecture

The proposed system has three stages to send real-time temperature alert call. First stage involves the installation of sensors inside the car to collect the kid and driver data inside the car. Second stage includes a microcontroller- Arduino MKR1000 to receive data from the sensors and then sends it to the Arduino IoT cloud for storing and analysis. Third stage of the system involves the integration of IFTTT web application, which is used to to trigger an call alert to the driver if the temperature inside the car reaches a certain level. By implementing these three stages, this system helps us to reduce the child death rate due to heat stores in smart car.

II. COMPONENTS AND SOFTWARE

A. Sensors

➢ *SW 420 Vibration Sensor:*

This vibration sensor has 3 pin (VCC, DO, GND) which is a is a high sensitivity non-directional vibration sensor with an potentiometer which controls the sensitivity of this sensor.Specifications are, voltage is 3.3V - 5V. Here, we are using this sensor to check the car engine status - on/off.

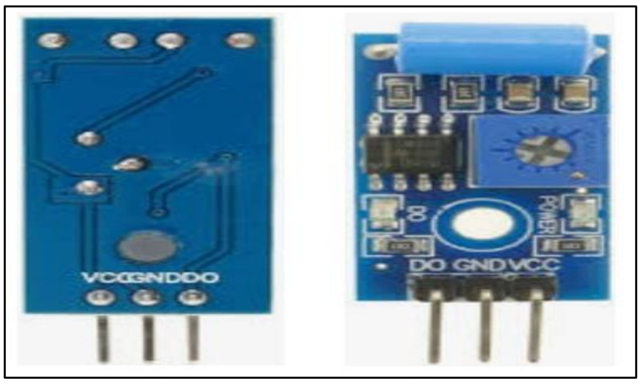


Fig 2: SW 420 Vibration Sensor

➢ *PIR Motion Sensor V1.2:*

This PIR motion sensor v1.2 has 4 pins (D1, NC, VCC, GND) which allows to sense motion, usually human movement in its range. Specifications of this sensor are, Voltage is 3v-5v, measuring range is 0.1m - 6m, default detecting distance is 3m, wavelength is 7-14um, detecting angle is 120 degrees and the holding time is 1-25s. Here we are using this PIR motion sensor, to sense the Kid in the back seat of car and to detect the driver in driver seat.

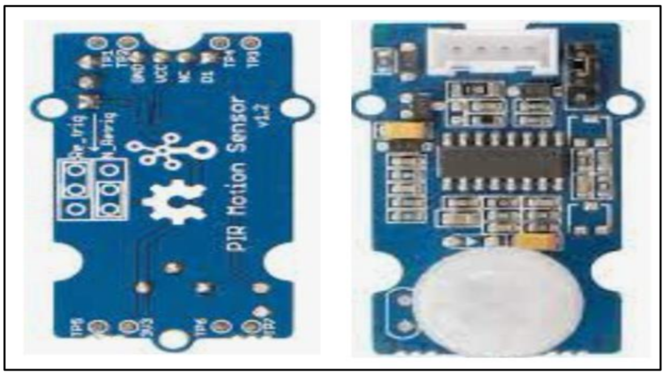


Fig 3: PIR Motion Sensor V1.2

➢ *Force Sensing Resistor*

This sensor has 2 pins and it gives a value when a pressure is applied. Output is totally depended on the area on sensors's surface when force is applied. Specifications of this FSR are, Voltage is 3.3V/5V, Force sensitivity range - 0.2N - 20N, Analog output - 0 to 650. Here we are using to to check the kid and driver press on seats.

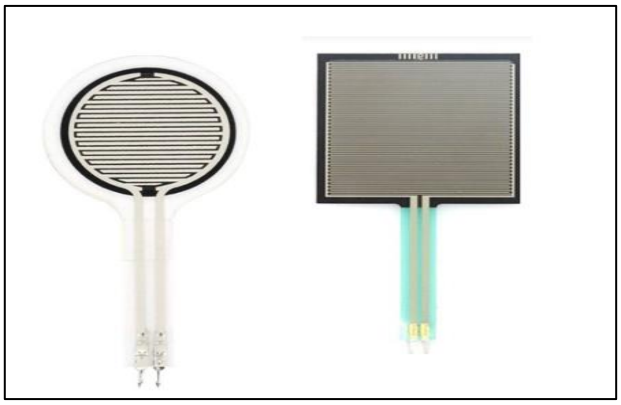


Fig 4: Force Sensing Resistor

➢ *Temperature Sensor V1.2*

This temperature sensor v1.2 has 4 pins (SIG, NC, VCC, GND) which has a in-built thermistor which detects the temperature.The resistance of a thermistor will increase when the temperature decreases. Specifications of this sensor are, Voltage is 3.3 ~ 5V, Zero power resistance is 100 KΩ, Resistance Tolerance is ±1%, Operating temperature range -40~ +125 ℃ and Nominal B-Constant 4250 ~ 4299K. Here we are using this temperature sensor to detect the temperature inside car.

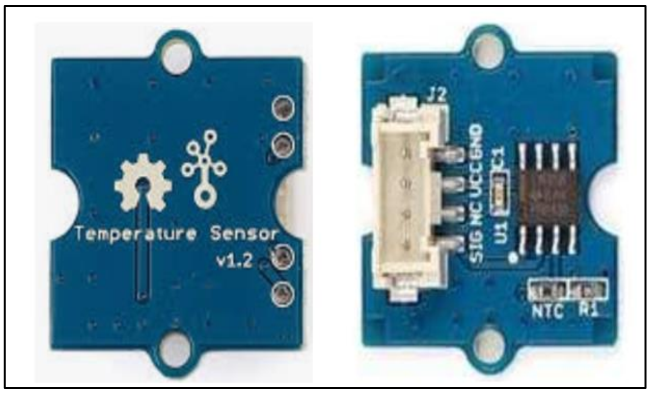


Fig 5: Temperature Sensor V1.2

B. Actuators

➢ *Buzzer*

This buzzer v1.2 has 4 pin (GND, VCC, NC, SIG) has a piezo buzzer as the main component. This can be connected to digital output and will emit a tone when there is output. Specifications are, voltage is 3.3V - 5V, and sound output is >= 85dB and resonant frequency is 2300+300 or -300HZ. Here we are using this buzzer inside the car to ring when there is fluctuations in temperature with a kid motion and no driver.

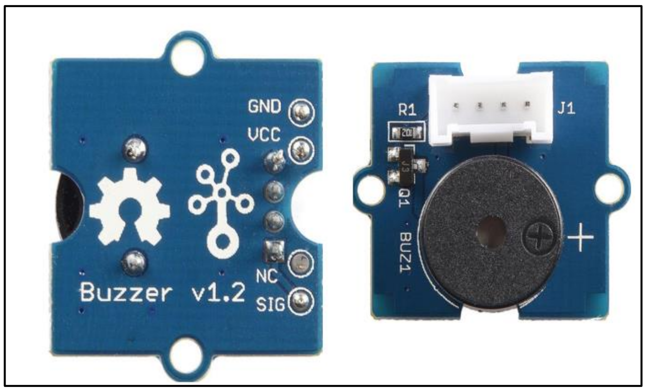


Fig 6: Buzzer

➢ *LED*

Light emitting diode is a semiconductor device that emits light when current flows through it. Here we are using these led's to represent the detection of kid and Driver.

Fig 7: LED

C. Micro Controller - Arduino MKR1000

This Arduino MKR1000 is based on Atmel SAMD21 microcontroller which features built-in Wi-Fi which makes it easy to connect to Internet and communicate with other devices. This has 14 digital input/output pins, 6 analog inputs and communication interfaces (SPI, I2C and UART). This can be connected using USB cable or with an Li-Po battery and this supports 3.3V- 5V. This is compatible with Arduino IoT cloud technology.

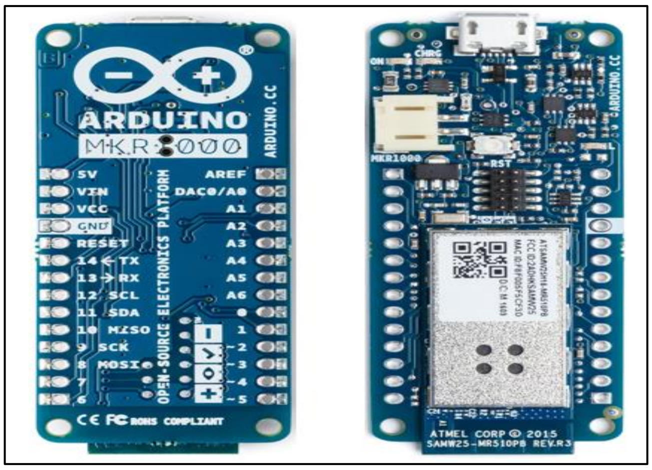


Fig 8: Micro Controller - Arduino MKR1000

D. Arduino IoT Cloud

Arduino IoT cloud is a cloud-based service that allows users to connect their devices to the internet and manage them remotely. It provides a platform for collecting and analyzing data from connected devices, creating custom dashboards, and setting up alerts and notifications. With the Arduino IoT cloud, it is easy to monitor and control devices from anywhere with an internet connection. For our proposed system, we have utilized the Arduino IoT cloud to collect and analyze data from the Arduino MKR1000 board and provide real-time monitoring and alerts for temperature fluctuations.

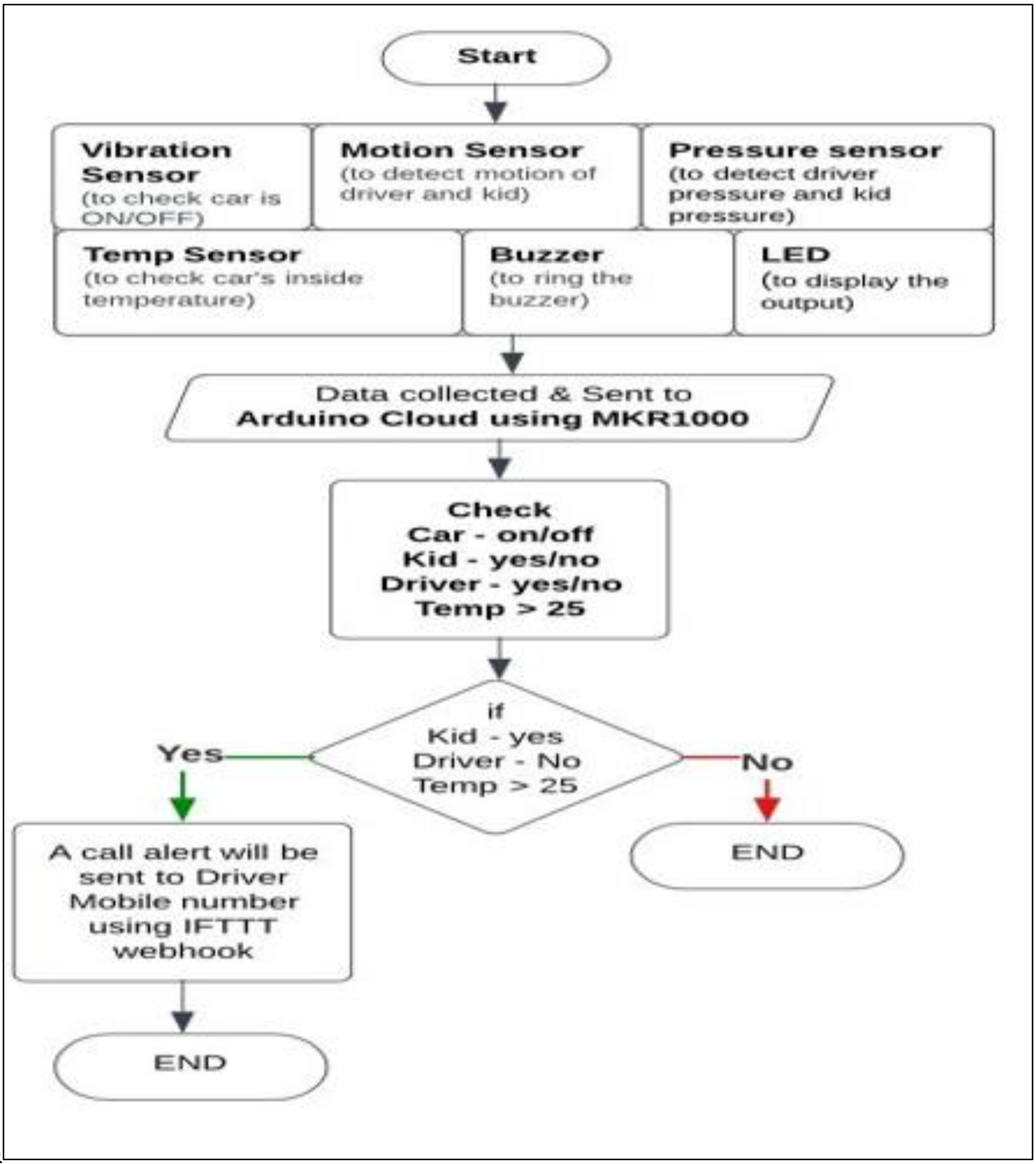
Fig 9: Arduino IoT Cloud

E. IFTTT Web Application

IFTTT (If This Then That) is a web-based service that allows users to create automated workflows between different web applications and services. With IFTTT, users can create custom applets that connect different web services, allowing for automated actions based on triggers. IFTTT can be used to integrate different devices and services and create custom triggers and actions such as calling, messaging, and e-mailing. This makes it easier to manage and control IoT devices and create more advanced and efficient automation workflows.

Overall, the importance of IFTTT lies in its ability to simplify and automate complex workflows, saving time and effort for users and enabling new levels of productivity and efficiency.

Fig 10: IFTTT Web Application


Start
Vibration Sensor (to check car is ON/OFF)
Motion Sensor (to detect motion of driver and kid)
Pressure sensor (to detect driver pressure and kid pressure)
Temp Sensor (to check car's inside temperature)
Buzzer (to ring the buzzer)
LED (to display the output)
Data collected & Sent to Arduino Cloud using MKR1000
Check Car - on/off Kid - yes/no Driver - yes/no Temp > 25
if Kid - yes Driver - No Temp > 25
Yes
No
A call alert will be sent to Driver Mobile number using IFTTT webhook
END
END

Q

Fig 11: Flowchart

This flowchart explains about the proposed system, from data collection to real-time alerts. The system collects data from the sensors installed in the car and then transmitted to the cloud for analysis. Based on the analysis, the system detects the child's presence and the temperature status inside the car. If there is no driver inside the car and the child's temperature exceeds a certain threshold, an alert is sent to driver mobile number using IFTTT webhook and applets. The system's workflow is designed to be efficient and reliable,ensuring the safety and well-being of children in smart cars.

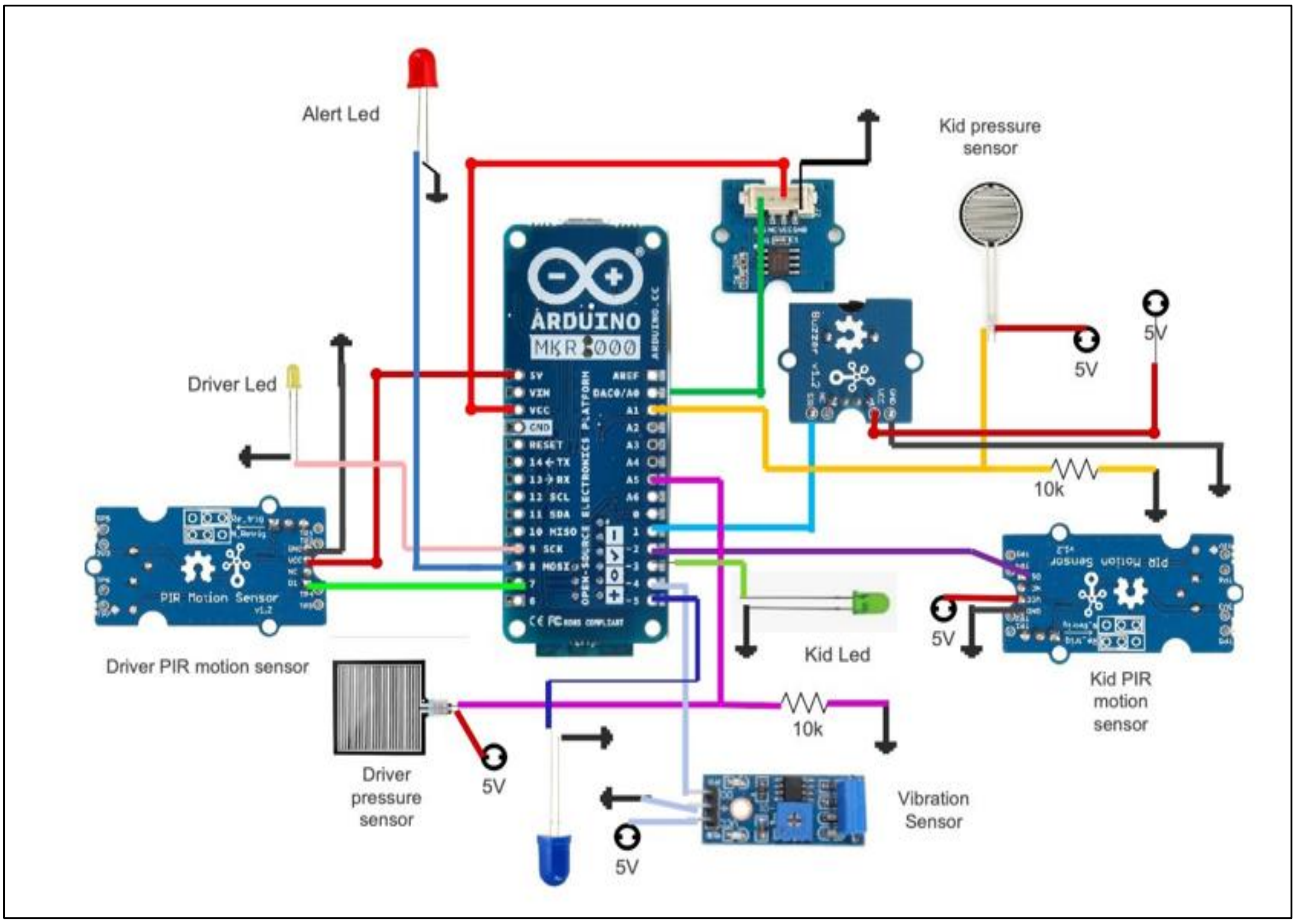


Fig 12: Schematic Diagram

Table 1: Circuit Pin Connections

MKR 1000	Vibration	Car Led	Kid motion	Kid pressure	Kid led	Driver motion	Driver pressure	Driver Led	Temp	Buzzer	Alert Led
5V	VCC	-	VCC	1 pin	-	VCC	1 pin	-	-	VCC	-
GND	GND	-ve	GND	-	-ve	GND	-	-ve	GND	GND	-ve
4 (D)	DO	-	-	-	-	-	-	-	-	-	-
D5	-	+ve	-	-	-	-	-	-	-	-	-
2 (D)	-	-	DI	-	-	-	-	-	-	-	-
A1	-	-	-	2 pin	-	-	-	-	-	-	-
3 (D)	-	-	-	-	+ve	-	-	-	-	-	-
7 (D)	-	-	-	-	-	DI	-	-	-	-	-
A5	-	-	-	-	-	-	2 pin	-	-	-	-
9 (D)	-	-	-	-	-	-	-	+ve	-	-	-
A0	-	-	-	-	-	-	-	-	SIG	-	-
1 (D)	-	-	-	-	-	-	-	-	-	SIG	-
8 (D)	-	-	-	-	-	-	-	-	-	-	+ve

➢ *Setting up the Circuit System in Car Setup*

We arranged a vibration sensor beside the car steering to check car engine status either ON/OFF. To check the driver, a motion sensor at the driver leg space, FSR (pressure) sensor on the driver seat. To check the child, motion sensor on the back of driver seat, child FSR (pressure) sensor on the child seat, Temperature, buzzer and led's are placed in-between of driver and child seat. Based on the conditions, if there is no driver and a child is inside the car with higher temperature then a buzzer will the ringed with an led light.

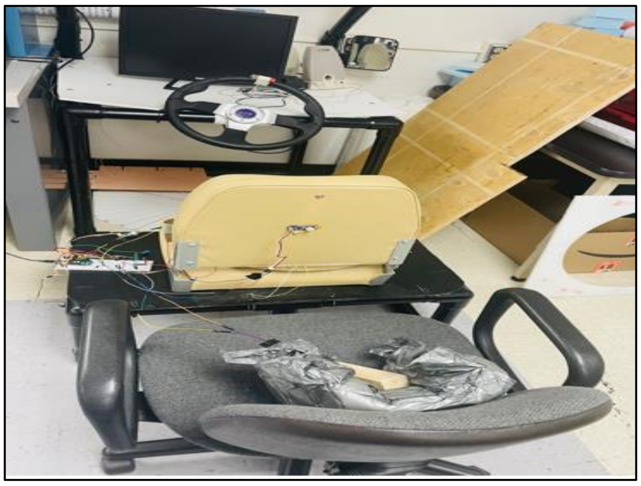
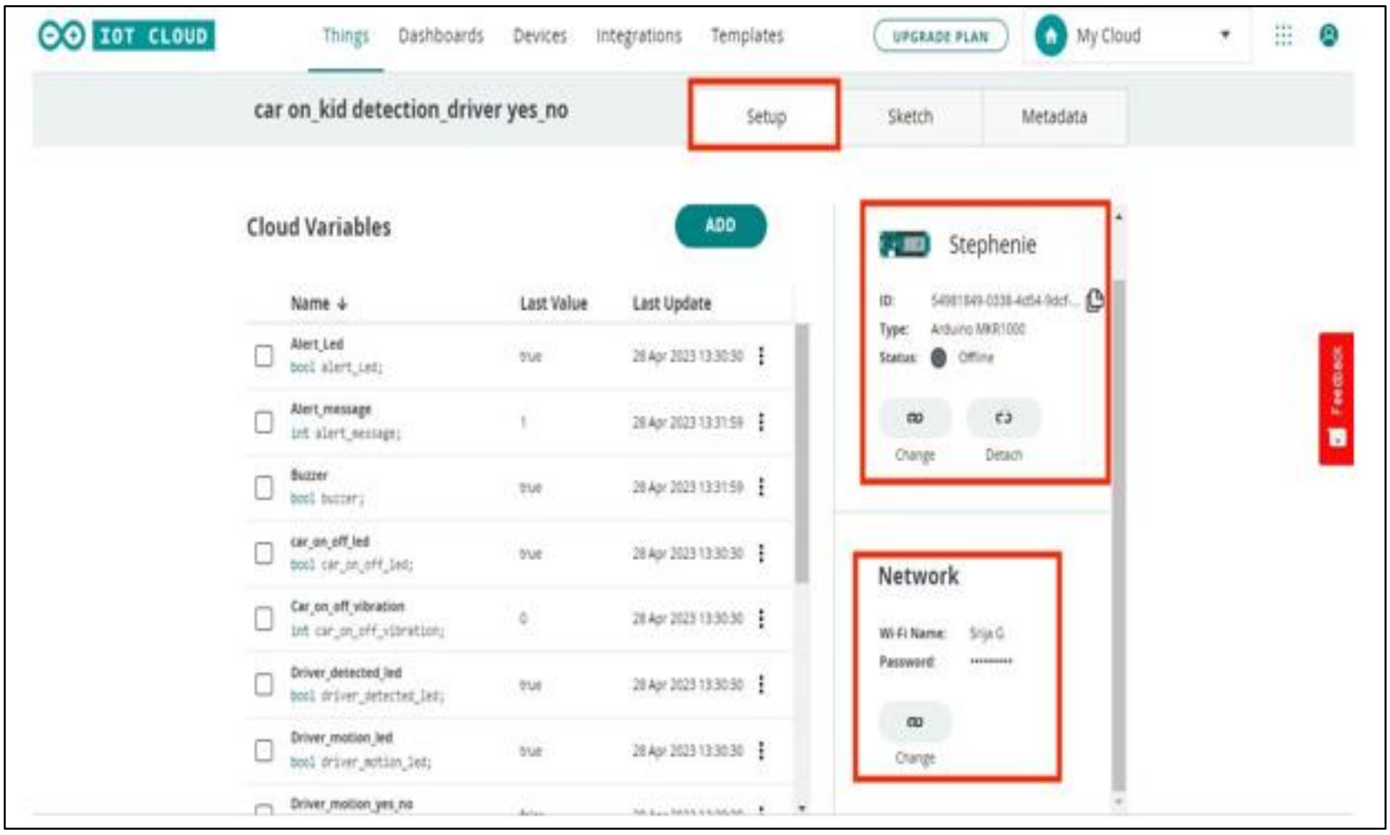


Fig 13: Circuit System in Car Setup

➢ *Sending and Storing the Sensor Data*

By using MKR1000 and Arduino IoT cloud, we have set the MKR device, wifi network, and the things(sensors) in the cloud website. Here, things represents a physical device which is connected inside the car to send and receive data. In order to get the sensor data, we need to add the the sensor data in variables which manages data associated with a Thing. These can be modified through the Cloud platform or through an API. Then we will associate the microcontroller (MKR) with cloud along with network.It's better to choose a personal network instead on a public network.



car on_kid detection_driver yes_no

Cloud Variables

Name ↓	Last Value	Last Update
Alert_Led bool alert_led;	true	28 Apr 2023 13:30:30
Alert_message int alert_message;	1	28 Apr 2023 13:31:59
Buzzer bool buzzer;	true	28 Apr 2023 13:31:59
car_on_off_led bool car_on_off_led;	true	28 Apr 2023 13:30:30
Car_on_off_vibration int car_on_off_vibration;	0	28 Apr 2023 13:30:30
Driver_detected_led bool driver_detected_led;	true	28 Apr 2023 13:30:30
Driver_motion_led bool driver_motion_led;	true	28 Apr 2023 13:30:30
Driver_motion_yes_no	[illegible]	[illegible]

Stephenie
ID: 54981849-0338-4d54-9dcf...
Type: Arduino MKR1000
Status: ● Offline

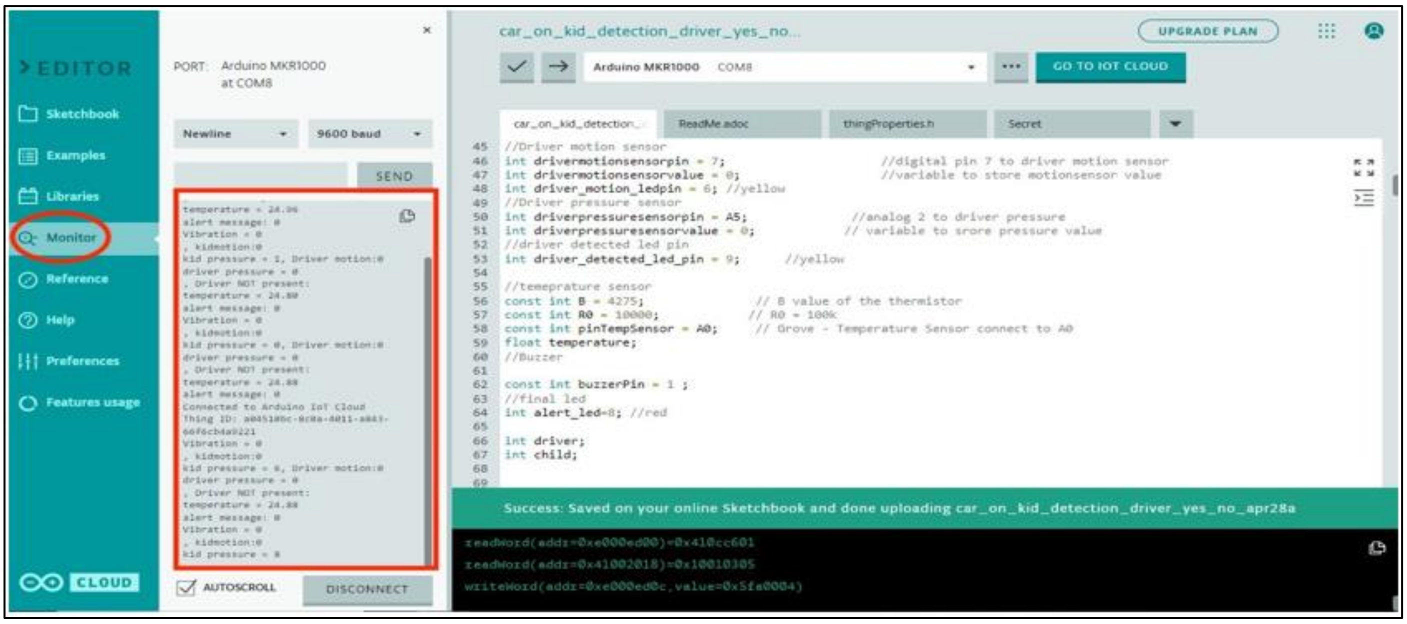
Network
Wi-Fi Name: Srija G
Password: *********

Fig 14: Sensor Data

Once we are done with setting up the devices, an auto generator code appeared on the sketch tab. Based on our proposed system and expected output, we have modified the code with necessary conditions and loops and then upload the code in the cloud. Here is our code, Arduino IoT Cloud code for 'Temperature call alert'

```
/*
  Sketch generated by the Arduino IoT Cloud Thing
"Untitled"
  https://create.arduino.cc/cloud/thing
s / a04510bc-8c0a-4011-a843-66f6cb4a9221

Arduino IoT Cloud Variables description
  The following variables are automatically generated
and updated when changes are made to the Thing

float temperature_inside_car; int
alert_message;
int car_on_off_vibration; int
driver_pressure;
int kid_pressure;
CloudLocation location;
bool alert_Led;
bool buzzer;
bool car_on_off_led;  bool
driver_detected_led;
bool driver_motion_yes_no; bool
kid_detected_led;
  Variables which are marked as READ/WRITE in
the Cloud Thing will also have functions
  which are called when their values are changed from
the Dashboard.
  These functions are generated with the Thing and
added at the end of this sketch.
*/
#include "thingProperties.h"
//vibration sensor
  int vibrationsensorpin = 4; //digital pin 2 to vibration
sensor
int vibrationsensorvalue = 0;  //variable to store vib value
int car_ledpin = 5;  //blue                    /digital pin 3 to
vibration
//kid motion sensor
  int kidmotionsensorpin = 2;             //digital pin 4 to
motion sensor
  int kidmotionsensorvalue = 0;          //variable to
store mot value
//pressure sensor
int kidpressuresensorpin = A1;                      //analog pin
1
  int kidpressuresensorvalue = 0;
                                                          //var
iable to store pressure sensor data
//kid detection led pin
  int kid_detected_led_pin = 3;        //green        / kid
detected signal
//Driver motion sensor
  int drivermotionsensorpin = 7;
                                                        //digit
al pin 7 to driver motion sensor
  int drivermotionsensorvalue = 0;
                                                          //var
iable to store motionsensor value
//Driver pressure sensor
  int driverpressuresensorpin = A5;
                                                         //ana
log 2 to driver pressure
  int driverpressuresensorvalue = 0;                  //
variable to srore pressure value
//driver detected led pin
int driver_detected_led_pin = 9;              //yellow
//temeprature sensor
const int B = 4275;                        // B value of the
thermistor const int R0 = 10000;    // R0 = 100k
  const int pinTempSensor = A0;          // Grove -
Temperature Sensor connect to A0
float temperature;
//Buzzer
const int buzzerPin = 1 ;
//final led
int alert_led=8; //red
int driver;
nt child;
void setup() {
/ Initialize serial and wait for port to open:
Serial.begin(9600);
pinMode(vibrationsensorpin,INPUT);
pinMode(car_ledpin,OUTPUT);
pinMode(kidmotionsensorpin,INPUT);
//pinMode(kidpressuresensorpin,  INPUT);
pinMode(kid_detected_led_pin,OUTPUT);
pinMode(drivermotionsensorpin, INPUT);
//pinMode(driverpressuresensorpin,INPUT);
pinMode(driver_detected_led_pin,OUTPUT);
pinMode(buzzerPin,OUTPUT);
pinMode(alert_led, OUTPUT);
  // This delay gives the chance to wait for a Serial
Monitor without blocking if none is found
delay(1500);
// Defined in thingProperties.h
initProperties();
// Connect to Arduino IoT Cloud
ArduinoCloud.begin(ArduinoIoTPreferredConnection);
/*
     The following function allows you to obtain
more information
         related to the state of network and IoT
Cloud connection and errors
the higher number the more granular information you'll
get.
The default is 0 (only errors). Maximum
is 4
*/
setDebugMessageLevel(2);
ArduinoCloud.printDebugInfo();
}
void loop()
{ ArduinoCloud.update();
// Your code here
/*
//reading vibration sensor data
vibrationsensorvalue = digitalRead(vibrationsensorpin);
//printing vibration values
Serial.print("vibration: ");
Serial.print(vibrationsensorvalue);
*/
//
long measurement =TP_init(); //vibration pin
delay(50);
Serial.print("Vibration = ");
Serial.println(measurement);
vibrationsensorvalue= measurement;

if (vibrationsensorvalue > 200)
{ digitalWrite(car_ledpin, HIGH);
car_on_off_led=car_ledpin;
delay(1000);
}
else
{ digitalWrite(car_ledpin, LOW);
}
//
//reading kid motion sensor data
kidmotionsensorvalue = digitalRead(kidmotionsensorpin);
  //printing motion values Serial.print(",
kidmotion:");
Serial.println(kidmotionsensorvalue);
//kid pressure sensor data
                    k i d p r e s s u r e s e n s o r v a l u
e

= analogRead(kidpressuresensorpin);
Serial.print("kid pressure = ");
Serial.print(kidpressuresensorvalue);
kid_pressure=kidpressuresensorvalue;
//kid detected led
if (kidmotionsensorvalue=1 && kidpressuresensorvalue
> 3)
{
digitalWrite(kid_detected_led_pin, HIGH);
child=1;
kid_detected_led=kid_detected_led_pin;
delay(100);  }
else{
digitalWrite(kid_detected_led_pin, LOW);
child=0;
kid_detected_led=kid_detected_led_pin;
delay(100);
}
//Driver motion values
    d r i v e r m o t i o n s e n s o r v a l u
e

= digitalRead(drivermotionsensorpin );
//printing motion values
Serial.print(", Driver motion:");
Serial.println(drivermotionsensorvalue);
//Driver pressure sensor data
    d r i v e r p r e s s u r e s e n s o r v a l u e
= analogRead(driverpressuresensorpin);
  Serial.print("driver pressure = ");
Serial.println(driverpressuresensorvalue);
driver_pressure=driverpressuresensorvalue;
//Driver detected led
          i f  ( d r i v e r m o t i o n s e n s o r v a l u e  ==  1  &
& driverpressuresensorvalue > 20)
{
Serial.println(", Driver present:"); driver=1;
digitalWrite(driver_detected_led_pin,    HIGH);
driver_detected_led=driver_detected_led_pin;
delay(100);
}
else{
Serial.println(", Driver NOT present:");
driver=0; digitalWrite(driver_detected_led_pin,
LOW); delay(100);
}
//temperature sensor
int a = analogRead(pinTempSensor);
float R = 1023.0 / a - 1.0;
R = R0 * R;
//float temp;
      temperature = 1.0 / (log(R / R0) / B + 1 /
298.15) - 273.15;
Serial.print("temperature = ");
Serial.println(temperature);
temperature_inside_car=temperature;
delay(2000);
//Buzzer
if (temperature > 25 && child == 1 && driver == 0)
{
          Serial.println("Child Present in car with HIGH
temperature ");
alert_message=1;

buzzer=HIGH;
digitalWrite(buzzerPin, HIGH);
delay(1000);
digitalWrite(buzzerPin, LOW);
digitalWrite(alert_led, HIGH);
alert_Led = alert_led;
delay(1000);
}


else
{
alert_message=0; buzzer=LOW;
digitalWrite(alert_led, LOW);
alert_Led=alert_led;
}
Serial.print("alert message: "); Serial.println(alert_message);
}
long TP_init(){ delay(10);
          long   measurement=pulseIn   (vibrationsensorpin,
HIGH);  //wait  for  the  pin  to  get  HIGH  and  returns
measurement
return measurement;
}
/*
        Since CarOnOffVibration is READ_WRITE
variable, onCarOnOffVibrationChange() is
       executed every time a new value is received from IoT
Cloud.
*/
void onCarOnOffVibrationChange() {
       // Add your code here to act upon
CarOnOffVibration change
}
/*
      Since KidMotionYesNo is READ_WRITE variable,
onKidMotionYesNoChange() is
       executed every time a new value is received from IoT
Cloud.
*/
void onKidMotionYesNoChange() {
        // Add your code here to act upon
KidMotionYesNo change
}
/*
        Since KidHeartbeatYesNo is READ_WRITE
variable, onKidHeartbeatYesNoChange() is
       executed every time a new value is received from IoT
Cloud.
*/
void onKidHeartbeatYesNoChange() {
        // Add your code here to act upon KidHeartbeatYesNo
change
}
/*
            Since KidPressure is READ_WRITE variable,
onKidPressureChange() is
        executed every time a new value is received from IoT
Cloud.
*/
void onKidPressureChange()  {
// Add your code here to act upon KidPressure change
}
/*
         Since DriverMotionYesNo is READ_WRITE
variable, onDriverMotionYesNoChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onDriverMotionYesNoChange() {
        // Add your code here to act upon
DriverMotionYesNo change
}
     /*Since DriverPressure is READ_WRITE variable,
onDriverPressureChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onDriverPressureChange()  {
// Add your code here to act upon DriverPressure change
}
/*
       Since DriverHeartbeatYesNo is READ_WRITE
variable, onDriverHeartbeatYesNoChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onDriverHeartbeatYesNoChange() {
       // Add your code here to act upon
DriverHeartbeatYesNo change
}
     /*Since AlertMessage is READ_WRITE variable,
onAlertMessageChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onAlertMessageChange()  {
// Add your code here to act upon AlertMessage change
}
     /*Since TemperatureInsideCar is READ_WRITE
variable, onTemperatureInsideCarChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onTemperatureInsideCarChange() {
       // Add your code here to act upon
TemperatureInsideCar change
}
     /*Since Buzzer is READ_WRITE variabl
e, onBuzzerChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onBuzzerChange()  {
// Add your code here to act upon Buzzer change
}
     /*Since KidMotionLed is READ_WRITE variable,
onKidMotionLedChange() is
        executed every time a new value is received from
IoT Cloud.
*/
void onKidMotionLedChange()  {
// Add your code here to act upon KidMotionLed change
}
     /*Since KidDetectedLed is READ_WRITE variable,
onKidDetectedLedChange() is

        executed every time a new value is received from IoT
Cloud.
*/
void onKidDetectedLedChange()  {
         // Add your code here to act upon KidDetectedLed
change
}
     /*Since DriverMotionLed is READ_WRITE variable,
onDriverMotionLedChange() is
        executed every time a new value is received from IoT
Cloud.
*/
void onDriverMotionLedChange()  {
         // Add your code here to act upon
DriverMotionLed change
}
     /*Since DriverDetectedLed is READ_WRITE variable,
onDriverDetectedLedChange() is
        executed every time a new value is received from IoT
Cloud.
*/
void onDriverDetectedLedChange()  {
         // Add your code here to act upon
DriverDetectedLed change
}
     /* Since TemperatureHigherLed is READ_WRITE
variable, onTemperatureHigherLedChange() is executed
every time a new value is received from IoT Cloud.
*/
void onTemperatureHigherLedChange() {
       // Add your code here to act upon
TemperatureHigherLed change
}
/*
     Since CarOnOffLed is READ_WRITE variable,
onCarOnOffLedChange() is
        executed every time a new value is received from IoT
Cloud.
*/
void onCarOnOffLedChange()  {
// Add your code here to act upon CarOnOffLed change
}
/*
     Since AlertLed is READ_WRITE variable,
onAlertLedChange() is
        executed every time a new value is received from IoT
Cloud.
*/
void onAlertLedChange()  {
// Add your code here to act upon AlertLed change
```

After updating code, we have verified and save the code on cloud. Later, uploaded the code (sketch) and then check the serial monitor for the output.



```
temperature = 24.96
alert message: 0
Vibration = 0
, kidmotion:0
kid pressure = 1, Driver motion:0
driver pressure = 0
, Driver NOT present:
temperature = 24.80
alert message: 0
Vibration = 0
, kidmotion:0
kid pressure = 0, Driver motion:0
driver pressure = 0
, Driver NOT present:
temperature = 24.88
alert message: 0
Connected to Arduino IoT Cloud
Thing ID: a0451a0c-8c8a-4011-a843-
66f6cb4a0221
Vibration = 0
, kidmotion:0
kid pressure = 0, Driver motion:0
driver pressure = 0
, Driver NOT present:
temperature = 24.88
alert message: 0
Vibration = 0
, kidmotion:0
kid pressure = 0
```

```
//Driver motion sensor
int drivermotionsensorpin = 7;                  //digital pin 7 to driver motion sensor
int drivermotionsensorvalue = 0;                //variable to store motionsensor value
int driver_motion_ledpin = 6; //yellow
//Driver pressure sensor
int driverpressuresensorpin = A5;             //analog 2 to driver pressure
int driverpressuresensorvalue = 0;           // variable to srore pressure value
//driver detected led pin
int driver_detected_led_pin = 9;      //yellow

//temeprature sensor
const int B = 4275;              // B value of the thermistor
const int R0 = 10000;           // R0 = 100k
const int pinTempSensor = A0;    // Grove - Temperature Sensor connect to A0
float temperature;
//Buzzer

const int buzzerPin = 1 ;
//final led
int alert_led=8; //red

int driver;
int child;
```

Success: Saved on your online Sketchbook and done uploading car_on_kid_detection_driver_yes_no_apr28a

```
readWord(addr=0xe000ed00)=0x410cc601
readWord(addr=0x41002018)=0x10010305
writeWord(addr=0xe000ed0c,value=0x5fa0004)
```

Fig 15: Cloud

Now, we have created the dashboard to view the connected devices in real time monitoring. While creating the dashboard with necessary things, we have linked the respective variable which we created earlier while setting the variables in things.

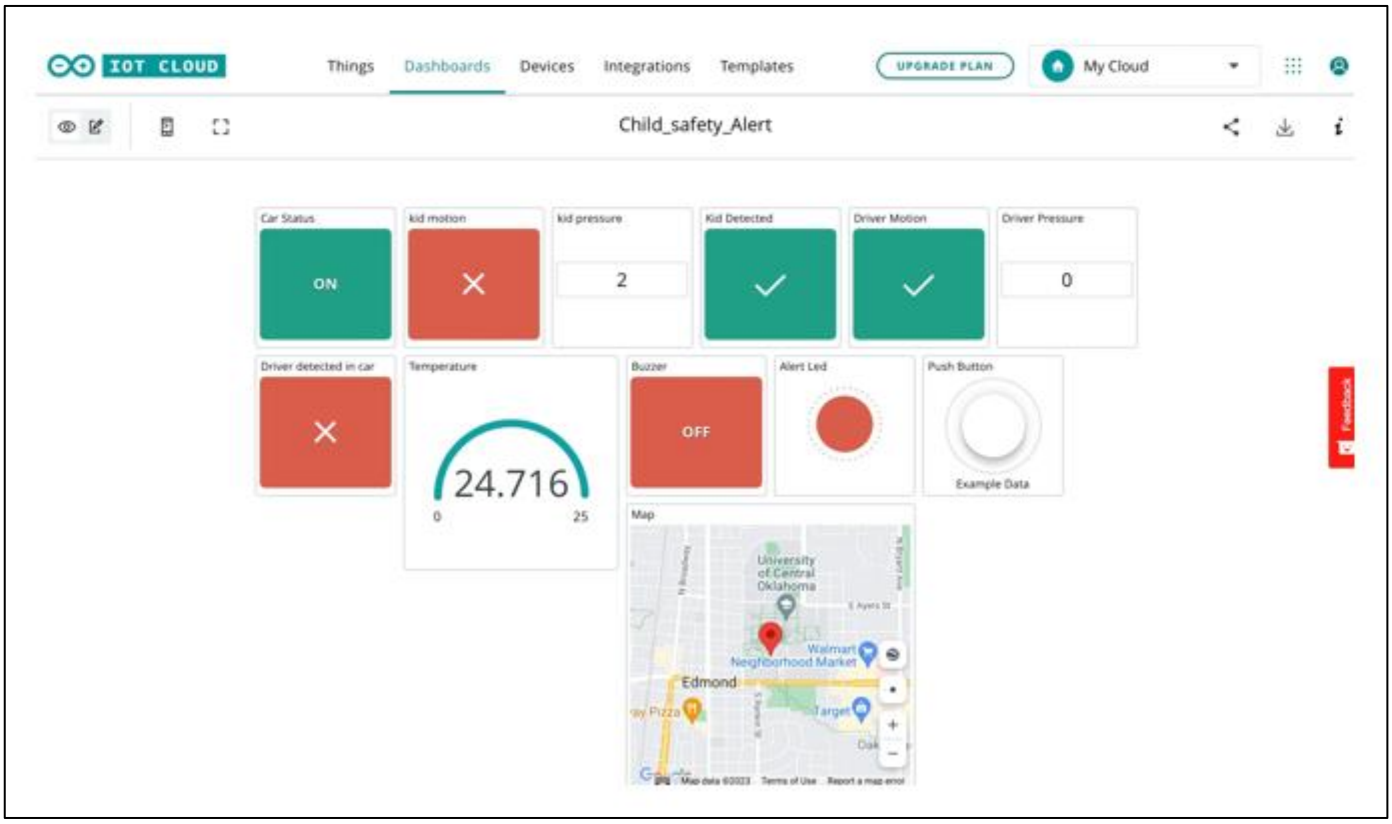
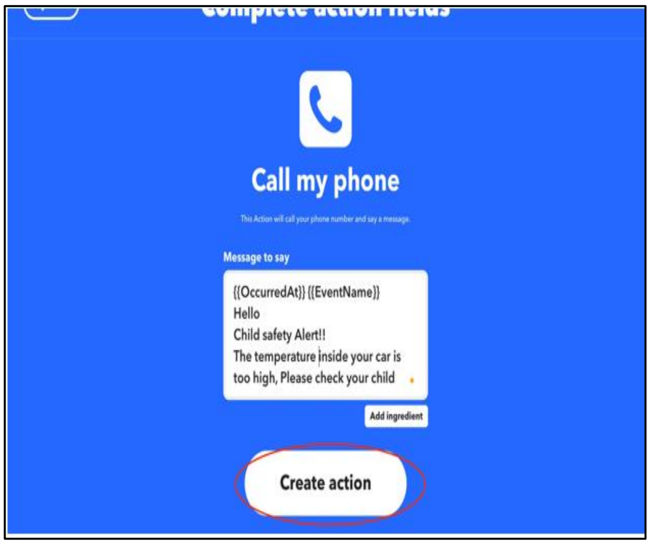


Fig 15: Dashboard

We have created an Applet by choosing the web request (call) option which is a pro feature of IFTTT and then updated the events and trigger along with the message.



{{OccurredAt}} {{EventName}}
Hello
Child safety Alert!!
The temperature inside your car is too high, Please check your child

Fig 16: Applet

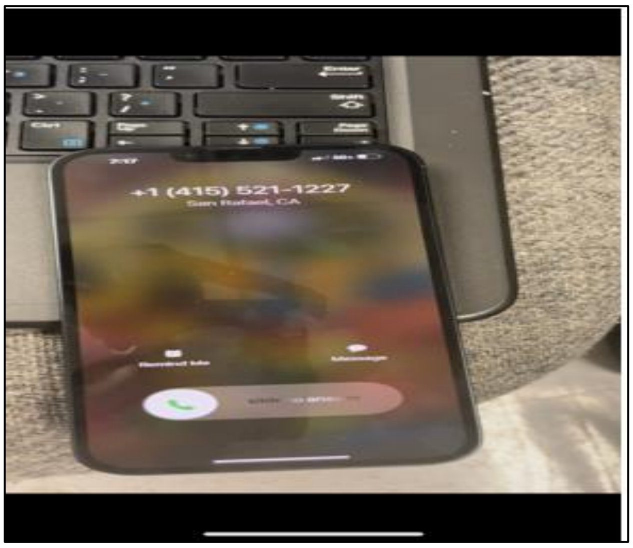


Fig 17: Mobile

executed every time a new value is received from IoT Cloud.
*/
void onLocationChange() {
// Add your code here to act upon Location change
}

Now, we need add the event name in the json key and copy that webhook url and paste that webhook link at the thing in Arduino cloud.Then a json event has been triggered and will get a call with the event name, date & time, and message.

III. RESULTS

While there is a child detection with our driver and temperature is too high inside car, able to get a temperature alert to the registered(driver) mobile number.

➢ *Simulation*

'NODE-RED' is one of the simulator tool which can be connected to our IoT projects. This flows to monitor, simulated production line, machine speed using MQTT and UMH data model. To create a flow and choose the nodes accordingly and set the filter node to 'block unless value changes'. To configure, choose switch mode which will check the value of messages which we have configured and passed through the function node. Now, we need to connect the nodes. Configured the mqtt- in node with the service name of hivemq 'testLocation/ DefaultProductionLine/Status/S'ateCurrent'. For mqtt-out node we need to select the broker with 'ia/factoryinsight/Aachen/ testAsset/State'. Our flow will be mqtt-in → json (json key) →Function → mqtt-out. As we are connecting this this to our Arduino, no need to write a new code, it will be using the same code.

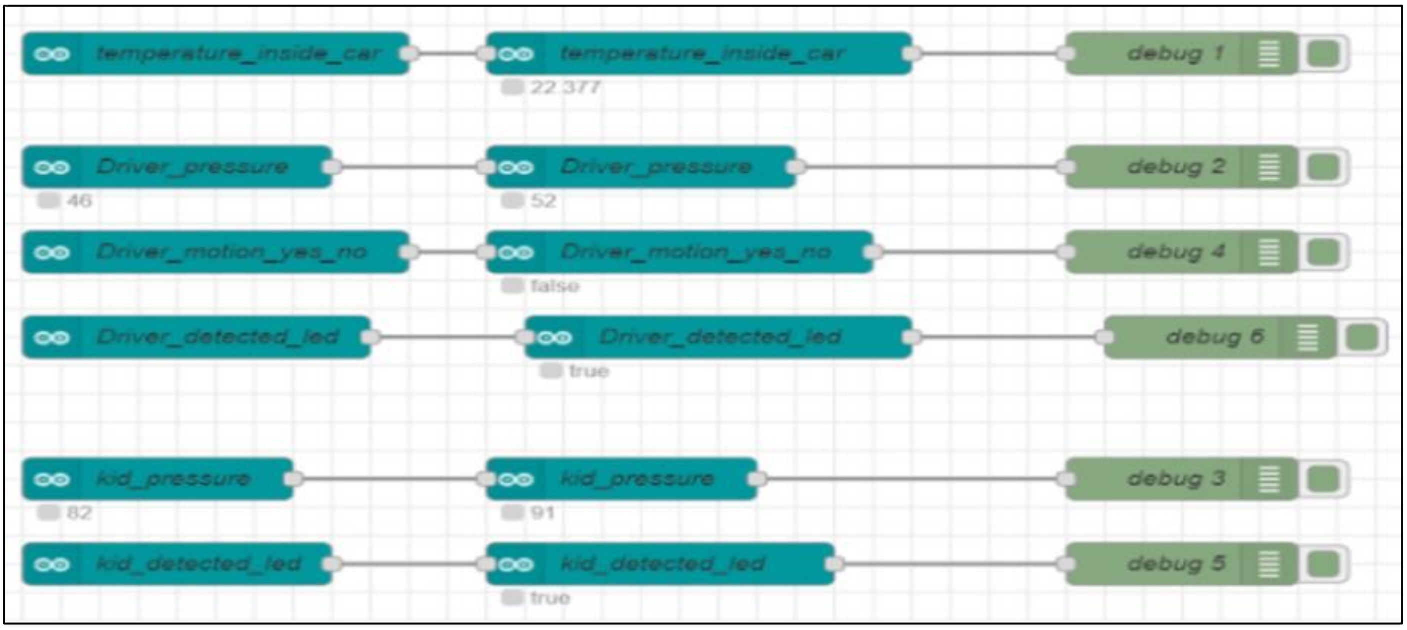


Fig 18: Simulation

IV. CONCLUSION

Numerous existing systems and prototypes have been developed to mitigate the occurrence of child heatstroke deaths, with several products available in the market. Based on my research work, most of the authors tried SMS option but not call service. So, our project aims to achieve this goal through a smartphone-based system with calling option not by SMS. By connecting our microcontroller to the cloud and retrieving data with minimal sensors. And we can run the system using any smartphone, which most people already possess.

REFERENCES

[1]. Please refer to this manual for Arduino cloud and IFTTT process.

[2]. Fatangare, M., Nimbalkar, A., Chite, G., Narkhede, A., & Khilnani, A. (2020). An Efficient Temperature Monitoring using Raspberry Pi. 2020 International Conference on Inventive Computation Technologies (ICICT). doi:10.1109/ icict48043.2020.9112376.

[3]. Chowdhury, U., Chowdhury, P., Paul, S., Sen, A., Sarkar, P. P., Basak, S., & Bhattacharya, A. (2019). Multi-sensor Wearable for Child Safety. 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). doi:10.1109/ uemcon47517.2019.8992950

[4]. Bhaskaran Venugopal, R., & Dudhe, R. (2021). IoT Based Advanced Heat Stroke Alarm System. 2021 International Conference on Computational Intelligence and KnowledgeEconomy(ICCIKE). doi:10.1109/ iccike51210.2021.941072610.1109/ ICCIKE51210.2021.9410726

[5]. Shi, D., Lu, J., Wang, J., Li, L., Liu, K., & Pan, M. (2020). No One Left Behind: Avoid Hot Car Deaths via WiFi Detection. ICC 2020 - 2020 IEEE International Conference on Communications (ICC). doi:10.1109/icc40277.2020.9148648

[6]. Xu, Q., Wang, B., Zhang, F., Regani, D. S., Wang, F., & Liu, K. J. R. (2020). Wireless AI in Smart Car: How Smart a Car Can Be? IEEE Access, 1–1. doi: 10 . 1109 /access.2020.297853

[7]. Moutaz Saleh., Fareeda Charkie., Rola Al-Hamad., Fatma Almisned. Designing a Smart Alarm System to Prevent Child Heatstroke in Vehicles. 2022 International Conference on Smart Systems and Power Management (IC2SPM) doi : 10.1109/ IC2SPM56638.2022.9988856

[8]. Barrera, J. P. S., Sandoval, G. M., Ortiz, G. C., Gonzalez, R. N., & Aguilar, E. R. (2014). A Multi-agent System to Avoid Heatstroke in Young Children Left in Baby Car Seats inside Vehicles. 2014 International Conference on Computational Science and Computational Intelligence. doi:10.1109/csci.2014.129

[9]. R. Lusso., M.Jensen., E. Walters., Ranger ., K. Alexander., AUTOMOBILE SAFETY - CHILD SEAT ENTRAPMENT AND MECHATRONIC WARNING SYSTEM. doi : 10.3182/20070820-3-US-2918.00040