Faculty of Engineering

GNG 1103

Hot Car Emergency

BabySafe by Arccode - User Guide

Team Members

Ekene Ebenebe

Kayden Wang

Nalida Awog-Badek

Nicole Meouch

Owen Haralovich

December 8, 2021

Abstract

The goal of this document is to highlight a user guide for the BabySafe device. This report covers the prototype design process, and the final design assembly.

Table of Contents

Table of Figures

Introduction

For this project, our client, Mansour Kharoub, is an engineer working in the United Arab Emirates (UAE). We have been tasked with designing a child monitoring system that alerts the guardian and passerby of a vehicle that a child is alone in the vehicle.

For our Prototype, we have addressed a major risk faced by an abandoned child in a vehicle, which is high temperatures. Our design is such that any interested individual(s) can follow through to replicate as well as improve on our solution. We had a budget of $50 to create our design and this document will be a guide as to how we balanced our budget, and navigated risks to create our design.

Relevant Softwares and Applications Used

Computer Aided Design

- Onshape: This is a cloud based CAD software. The casing for our design was created on here. A link to the CAD files is attached to the Appendix of this document.
- Inkscape and Adobe Pdf: Inkscape is a free and open-source vector graphics editor. It was used to format our CAD drawings, and casing engravings to Laser-cut graphics. Adobe pdf was an interface between Inkscape and the Laser cutting machine Used for our design.

Simulations and Functionality

- TinkerCad: This is a free online modelling program. The code for our design programming was tested on here using virtual components to run simulations to fine tune our design goals.
- Arduino: This is an open source electronic prototyping platform. We used an Arduino Nano as the microcontroller running our design logic, so we used the Arduino IDE to upload some already tested out programs to the microcontroller. Also, we utilised the vast code library available on Arduino to maximize functionality of our design components.

Bill of Materials

Item #	Item Description	Quantity	Unit Price (CAD)	Amount	Source
1	Arduino Nano	1	$8.00	$8.00	https://makerstore.ca/shop
2	NTC Thermistor	1	$2.18	$2.18	https://makerstore.ca/shop
3	PIR Sensor	1	$3.00	$3.00	https://www.digikey.ca/
4	Resistor set (8x220ohm, 1x100kohm)	27	$0.01	$0.27	https://makerstore.ca/shop
5	LED	8	$0.30	$2.40	https://makerstore.ca/shop
6	5ft of wire Length	1	$2.50	$2.50	https://makerstore.ca/shop
7	protoboard	2	$0.50	$1.00	https://makerstore.ca/shop
8	9v Rechargeable Battery	1	$2.98	$2.98	https://www.walmart.ca
9	Push button	1	$0.71	$0.71	https://makerstore.ca/shop
10	Mono enclosed Speaker	1	$4.50	$4.50	https://makerstore.ca/shop
11	PAM8302 Class D Audio Amplifier	1	$5.37	$5.37	https://www.digikey.ca/
12	9v battery connector	2	$1.59	$3.18	https://www.digikey.ca/
13	HC-SR04 Ultrasonic Sensor	1	$5.06	$5.06	www.robotshop.com/ca/en
14	1/4" thick 12" x 24" MDF board	1	$3.50	$3.50	https://makerepo.com
Total Cost				$44.65	

Figure 1: Bill of Materials

Prototype review and Replication

Design functionality flow chart

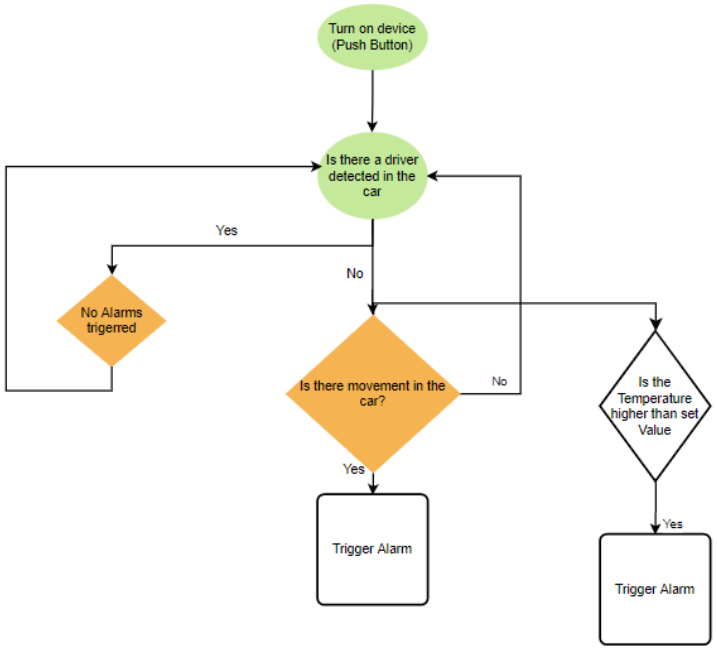

Figure 2: Functionality Flow Chart

Prototype Model views and Features

Figure 3: Top View of Prototype

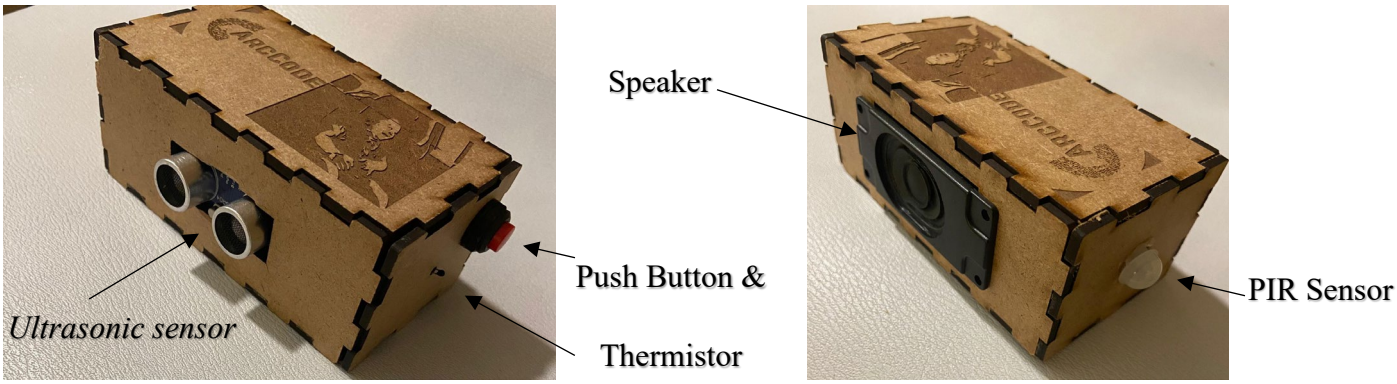

Figure 4: 3d View of Prototype

Prototype Features

- Detect motion
- Detect temperature change
- Trigger Alarm

Programming design Library and codes

All codes used are attached and tagged in the appendix of this document. They are split into the individual components that represent the features of our project, and a final code which is a network of all the components to function as a unified device.

Component Circuitry

For this project, some components will not be available on tinker cad, but substitutes will be used for simulation purposes.

Arduino nano pins and component assignment

- A0 => Thermistor
- D3 => Amplifier A+
- D8 => PIR Sensor
- D10 => Ultrasonic Sensor Echo
- D12 => Ultrasonic Sensor Trigger

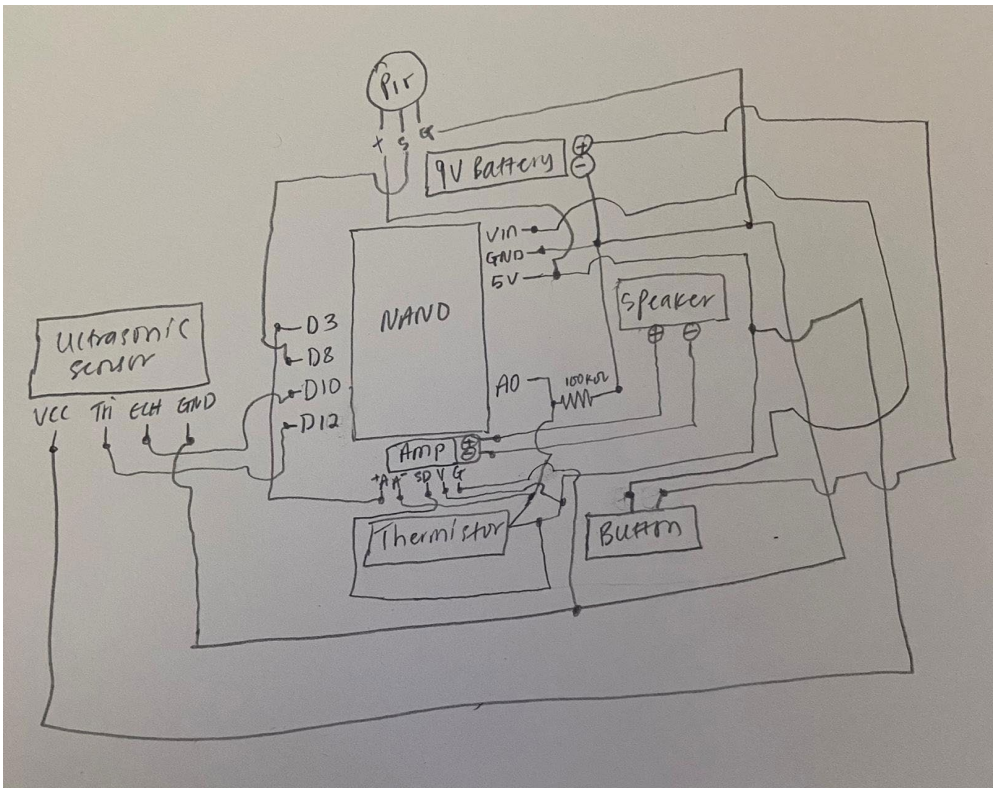

Figure 5: Circuitry

Part Production and Assembly

Onshape link shows access to the casing part file which includes a drawing that can be exported as a dxt file, then imported into Inkscape. Once on Inkscape, go to the top menu and click on fill

and stroke, click on stroke style, and change the width to 0.001mm. you'll notice the drawing seems to disappear, but this is normal as it's the setting for a vector cut to cut out the casing. After setting the stroke style, save as adobe pdf and ensure the resolution for rasterization is set at 600dpi. Then you're ready to laser cut. For the engravings, already edited pdf pics are attached to this document as well as other relevant docs available on the maker repository attached in the appendix of this document.

Soldering

Beginner level soldering skills are adequate for this project. Ensure that the heat is kept far from the components especially the PIR sensor. Best fit is advised for soldering, so it's important to measure wires and place on intended location before actual soldering. Also do not solder the PIR, Ultrasonic sensor and thermistor. Use Male and Female jumper cables to interface between Soldered wires and components. This helps for easy troubleshooting with the Arduino.

Box Assembly

Refer to hand sketch in fig. 5 to get an idea of component placement. Laser cut boxes are usually a tight fit but, in a case, where gluing is needed, leave access to the Arduino nano usb connector for easy trouble shooting as well

Conclusion

This was an exciting project to complete, most of the tasking work was in the soldering and troubleshooting, so its important to always measure first before machining. Overall, this prototype was well received by the client. Possible improvements include; Application to sync with the device for a better user experience, this will involve connecting additional components to the Arduino pins and resizing the box to fit everything. All links provided are easily edited to suit future needs.

Appendix

Onshape Link:

https://cad.onshape.com/documents/0029f90d0566c2041caa784e/w/dee985985cd52e04268a2816/e/a0bbec286821da951b344090

Maker repo Link:

https://makerepo.com/Ekene/1058.hot-car-emergency-babysafe-by-arcode

Code Library:

Speaker Tones

```
#include "pitches.h"

// notes in the melody:

int Emergencymelody[] = {NOTE_C4,};//Hot Car Emergency Tone

// note durations: 4 = quarter note, 8 = eighth note, etc.:

int noteDurations[] = {

 4, 8, 8, 4, 4, 4, 4, 4

};

void setup() {

  // iterate over the notes of the melody:

 Serial.begin(9600);

}

void loop() {

  // no need to repeat the melody.

   for (int thisNote = 0; thisNote < 1; thisNote++)

   {

    // to calculate the note duration, take one second divided by the note type e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.

    int noteDuration = 1000 / noteDurations[thisNote];

    tone(3, Emergencymelody[thisNote], noteDuration);

    int pauseBetweenNotes = noteDuration * 1.30;// to distinguish the notes, set a minimum time between them. the note's duration + 30% seems to work well:

    delay(pauseBetweenNotes);

    // stop the tone playing:

    noTone(3);  }}
```

PIR Code

```
int ampOff = 4;//AMP Shutdown
int PirAlarm = 3;//Speaker buzzer pin
//pir
int Pirsensor = 8;//the pin that the sensor is attached to
int PirState = LOW;//default to show no motion detected
int sensorvalue;//to store sensor status
void setup()
{
 pinMode(ampOff, OUTPUT);
 digitalWrite(ampOff, LOW);
 pinMode(Pirsensor, INPUT);
 Serial.begin(9600);
}
void loop()
{
 //pir
 sensorvalue = digitalRead(Pirsensor);//read value on sensor
 if (sensorvalue == HIGH)// check if the sensor is HIGH
 {
  digitalWrite(PirAlarm, HIGH);   //turn speaker ON
  Serial.println("Motion detected!");
  delay(500);                // delay 500 milliseconds
 }
 else
  {
  digitalWrite(PirAlarm, LOW); //turn speaker OFF
  delay(500);// delay 500 milliseconds
  digitalWrite (ampOff, HIGH);
  Serial.println("Car is empty!");
    }}
```

Ultrasonic Sensor Code

```
// Ultrasonic Sensor Code
const int trigPin = 12;
const int echoPin = 10;
// defines variables
long duration;
int distance;
```

```
void setup()
{
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
void loop()
{
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delay(500);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  if (distance >= 40)
  {
    Serial.print("Distance = ");
    Serial.println(distance);
    Serial.println("Driver out of car, check for movement and temperature!!!!!");
  }
  else
  {
    Serial.print("Distance: ");
    Serial.println(distance);
    Serial.println("Driver Inside car, all good");
  }}
```

Thermistor Code

```
//Thermistor code
int ThermistorPin = A0;
int Vo;
float R1 = 100000;
float logR2, R2, T;
```

```
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
void setup()
{
 //Thermistor Code
 pinMode(8, OUTPUT);//assign led to pin 8
 Serial.begin(9600);
}
void loop()
{
 //Thermistor Code
 Vo = analogRead(ThermistorPin);
 R2 = R1 * (1023.0 / (float)Vo - 1.0);
 logR2 = log(R2);
 T = (1.0 / (c1 + c2 * logR2 + c3 * logR2 * logR2 * logR2));//Temp in kelvins
 T = T - 273.15;//Temp in degree celcius
 // T = (T * 9.0) / 5.0 + 32.0;
 if (T > 28)
 {
  digitalWrite(8, HIGH); //turn on led
  delay(500);//delay for one second
 }
 else
 {
  digitalWrite(8, LOW); //turn off led
 }
 Serial.print("Temperature: ");
 Serial.print(T);
 Serial.print('\n');
 // Serial.println(" F");
 delay(1000);
}
```

Full System Code

```
#include "pitches.h"
#include <Buzzer.h>
// Ultrasonic Sensor Code
const int trigPin = 12;
const int echoPin = 10;
```

```
long duration;
int distance;
//Speaker code
int melody[] = {NOTE_A4, NOTE_B4, NOTE_C3, NOTE_A4, NOTE_B4, NOTE_C3}; // notes in the melody
int thisNote; int noteDuration; int pauseBetweenNotes;
int noteDurations[] = { 4, 8, 8, 4, 4, 4, 4, 4};  // note durations: 4 = quarter note, 8 = eighth note, etc.:,
//PIR Code
int Pirsensor = 8; //the pin that the sensor is attached to
int PirState = LOW; //default to show no motion detected
int sensorvalue; //to store sensor status
//Thermistor Code
int ThermistorPin = A0;
int Vo;
float R1 = 100000;
float logR2, R2, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;
void setup()
{
 Serial.begin(9600);
 //Ultrasonic sensor code
 pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
 pinMode(echoPin, INPUT); // Sets the echoPin as an Input
 //Speaker Code
 for (thisNote = 0; thisNote < 8; thisNote++) // iterate over the notes of the melody:
 {
  // to calculate the note duration, take one second divided by the note type.
  //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
  noteDuration = 1000 / noteDurations[thisNote];
  // to distinguish the notes, set a minimum time between them, the note's duration + 30% seems to work well:
  pauseBetweenNotes = noteDuration * 1.30;
 }
 //PIR Code
 pinMode(Pirsensor, INPUT);
}
void loop()
{
 //Ultrasonic Sensor Code
 digitalWrite(trigPin, LOW);// Clears the trigPin
```

```
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);// Sets the trigPin on HIGH state for 10 micro seconds
delay(500);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);// Reads the echoPin, returns the sound wave travel time in microseconds
distance = duration * 0.034 / 2;  // Calculating the distance
//PIR Code
sensorvalue = digitalRead(Pirsensor);//read signal from sensor
//Thermistor Code
Vo = analogRead(ThermistorPin);
R2 = R1 * (1023.0 / (float)Vo - 1.0);
logR2 = log(R2);
T = (1.0 / (c1 + c2 * logR2 + c3 * logR2 * logR2 * logR2));//Temp in kelvins
T = T - 273.15;//Temp in degree celcius
//Use Ultrasonic sensor to activate system
if (distance <= 20) //reverse is the case, but i am adjusting for the Ultrasonic sensor sensitivity
{
 Serial.print("Distance = ");
 Serial.println(distance);
 Serial.println("Driver out of car, check for movement and temperature!!!!!");
 if (sensorvalue == HIGH)// check for movement inside the car
 {
  //Turn Speaker ON
  tone(3, melody[thisNote], noteDuration);
  delay(pauseBetweenNotes);
  noTone(3); noTone(3); // stop the tone playing:
  Serial.println("Motion detected!");
  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.print('\n');
  delay(500);
 }
 //Multiconditional code, to check that there's movement and the car is hot.
 if (sensorvalue == HIGH && T > 24)// check if there's movement and high temperature   {
  //Turn Speaker ON
  tone(3, melody[thisNote], noteDuration);
  delay(pauseBetweenNotes);
```

```
    noTone(3); noTone(3); // stop the tone playing:
    Serial.println("Motion detected! Hot Car Emergency!!!");
    Serial.print("Temperature: ");
    Serial.print(T);
    Serial.print('\n');
    delay(500);                // delay 500 milliseconds
  }  }
else
{
  Serial.print("Distance: ");
  Serial.println(distance);
  Serial.println("Driver Inside car, all good");
  delay(500);
}
delay(1000);
```