دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

پروژه‌ی درس آزمایشگاه سخت‌افزار

عنوان:

مستند نهایی پروژه‌ی شناسایی موجود زنده در شب برای اتومبیل

نگارندگان:

علیرضا شاطری، رضا امینی

استاد گرامی:

جناب آقای دکتر اجلالی - جناب آقای دکتر فصحتی

زمستان ۱۴۰۱


بسم الله الرحمن الرحیم

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

محصول نهایی این پروژه، یک سیستم دید در شب است که درون اتومبیل قرار می‌گیرد و به راننده در هنگام رانندگی در تاریکی، کمک به سزایی می‌کند. در این سیستم اطلاعات از طریق یک دوربین حرارتی به ماژول رزپری منتقل می‌شود و کدهایی که در رزپری قرار داده شده است با انجا پردازش تصویری ساده، تشخیص خواهد داد که آیا موجود زنده‌ای در میدان دید راننده حضور دارد یا خیر. همچنین از طریق چراغ و صدا نتیجه را به راننده اطلاع می‌دهد.

به صورت دقیق‌تر، این محصول با کمک یک دوربین مادون قرمز، می‌تواند دمای موانع در سر راه راننده را از فاصله‌ی دور تشخیص دهد. سپس اگر دمای قسمتی از تصویر روبه‌رویش نسبت به دمای محیط به مقدار نسبتا قابل ملاحظه‌ای بالاتر باشد، سیستم متوجه حضور یک موجود زنده شده و شروع به هشدار دادن به راننده می‌کند. نکته‌ای که وجود دارد این است که اگر این تغییر دما آن قدر بالا باشد که دیگر نتواند به عنوان دمای واقعی بدن یک موجود زنده در نظر گرفت، آنگاه سیستم نیز موجود زنده‌ای را شناسایی نمی‌کند زیرا تنها محدوده‌ی دمایی خاصی‌ست که می‌توان مربوط به دمای بدن موجودات زنده باشد.

مزیت رقابتی اصلی این محصول هزینه‌ی پایین ساخت آن است که با تغییر در بعضی از ماژول‌های سیستم، حتی می‌توان به هزینه‌ی کمتر نیز رسید. همچنین محصول نهایی بسیار کوچک خواهد بود زیرا به جای چراغ‌هایی که در نمونه‌ی اولیه‌ی ما استفاده شده است، در واقع باید چراغ‌های خود اتومبیل قرار بگیرد و چون روشن کردن این چراغ‌ها از طریق ارتباط با کامپیوتر ماشین ممکن است در نتیجه کافی‌ست

پس از انجام پردازش‌ها توسط رزپری و سنسورها، دستور روشن شدن چراغ‌ها را به کامپیوتر اتومبیل ارسال و آن‌ها را روشن کرد.

فصل ۲

دیتاشیت محصول

محدوده‌ی دمای قابل استفاده	$-۱۰°C - ۷۵°C$
ولتاژ ورودی	۵ ولت
جریان ورودی	۲ آمپر
ابعاد	$۱۵cm * ۱۰cm * ۵cm$
وزن	۴۰۰ گرم
محدوده‌ی دمایی تشخیص موجود زنده	$۳۰°C - ۵۰°C$

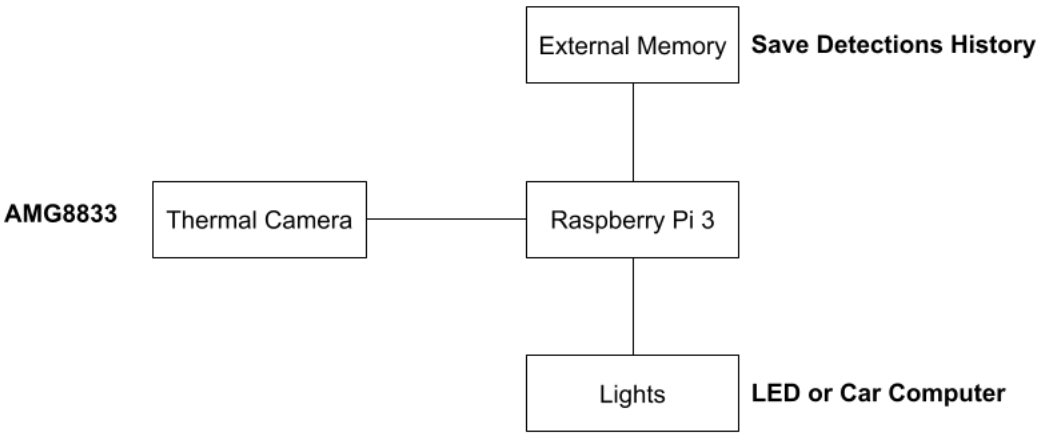
- وزن محصول بسته به شرایط و مواد مورد استفاده در تولید جعبه‌ی آن متغیر است.

فصل ۳

معماری سیستم

سیستم طراحی شده از ۲ قسمت اصلی ساخته شده است. قسمت سخت افزاری که شامل رزپری و سنسور حرارتی و مدارها می‌شود. قسمت نرم افزاری نیز که شامل پیاده‌سازی نرم افزاری است که در رزپری پای اجرا شده و توابع قسمت مختلف را مدیریت می‌کند.

معماری سطح بالای سیستم در شکل ۳-۱ قابل مشاهده است.



شکل ۳-۱: معماری سطح بالای سیستم

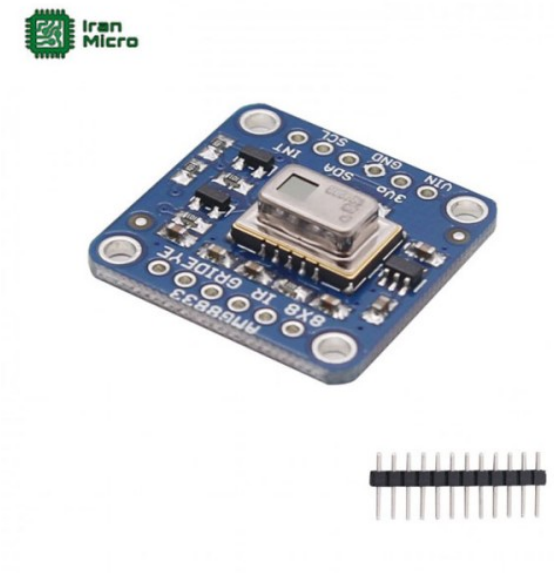
۳-۱ طراحی و پیاده‌سازی

اصلی‌ترین قسمت این پروژه، طراحی و پیاده‌سازی قسمت‌های سخت‌افزاری آن است. در زیر لیستی از قطعات سخت‌افزاری مورد استفاده آمده است و پس از آن توضیحاتی در مورد هر یک از سنسورها و نحوه کارکرد و راه‌اندازی آن ذکر شده است.

- برد Raspberry Pi 3
- سنسور دوربین حرارتی AMG8833
- چراغ‌های led

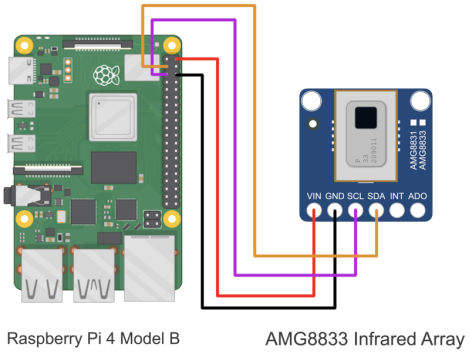
۳-۱-۱ دوربین حرارتی

یک آرایه سنسور مادون قرمز کم هزینه است که توسط پاناسونیک توسعه یافته است. برای استفاده با میکروکنترلرها در یک ماژول با شیفترهای سطح و تنظیم کننده ولتاژ یکپارچه شده است که برق و داده ۳ تا ۵ ولت را می دهد.
این سنسور تنها ۶۴ پیکسل (۸×۸) دارد که خیلی زیاد نیست اما برای آزمایش کافی و کار با آن ساده است، همچنین قیمت مناسبی نیز دارد.
ماژول را می‌توان به راحتی به برد متصل کرد و داده‌های دمایی تصویر را دریافت و پردازش نمود.



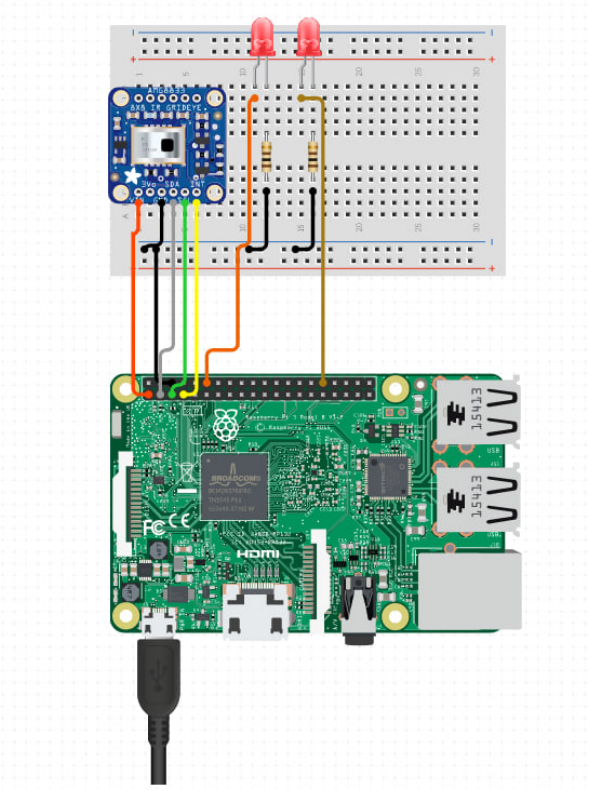
شکل ۳-۲: AMG8833

در تصویر زیر نیز می‌توانید نحوه‌ی اتصال این ماژول به رزپری را مشاهده کنید.



شکل ۳-۳: اتصال سنسور AMG8833

همچنین در تصویر زیر می‌توانید یک شماتیک کلی از نحوه‌ی اتصال کامل مدار ببینید.



شکل ۳-۴: شماتیک کلی از مدار

برای خواندن مقادیر از کتاب‌خانه‌ی smbus [۱] استفاده شده است. البته این کتابخانه مخصوص این ماژول نمی‌باشد و صرفا خواندن سریال از طریق i2c را برایمان راحت کرده است. در نتیجه کد اصلی خواندن دیتا دز دوربین حرارتی پیاده سازی شده است که در ادامه خواهید دید.

بخش‌های مهم پیاده‌سازی مربوط به این قسمت را می‌توانید در زیر مشاهده کنید. (برای جزئیات بیشتر به کد اصلی موجود در ریپازیتوری گیتهاب مراجعه کنید)

```
import smbus  # i2c bus

class i2c_driver(object):

    # Here is the most important function that reads serial
    # data from amg8833 sensor in little_endian order.

    def read16(self, register, little_endian=True):
```

[۱] https://pypi.org/project/smbus2/

```
        # read 16-bits from specified register
        result = self._bus.read_word_data(self._address, register) & 0
    xFFFF
        if not little_endian:
            result = ((result << 8) & 0xFF00) + (result >> 8)
        return result

    # ...

class AMG8833(object):
    # This important function gets the serial data and converts it
    # to temperatures in Centigrades. The return value of this function
    # will be an 8x8 matrix.

    def read_temp(self, PIXEL_NUM):
        T_arr = []  # temp array
        status = False  # status boolean for errors
        for i in range(0, PIXEL_NUM):
            raw = self.device.read16(GE_PIXEL_BASE + (i << 1))
            converted = self.twos_compl(raw) * 0.25
            if converted < -20 or converted > 100:
                return True, T_arr  # return error if outside temp window
            T_arr.append(converted)
        return status, T_arr
```

در نهایت تابعی که برای خواندن کل آرایه ۸ در ۸ نهایی استفاده می‌شود، تابع read_temp از کلاس تعریف شده در این کد است. این تابع آرایه‌ای شامل ۶۴ عدد integer برمی‌گرداند که هر کدام از این اعداد دمای یک پیکسل از ۶۴ پیکسل قابل دید توسط دوربین را نمایش می‌دهد. سپس این آرایه به کمک numpy به صورت یک مارتیس ۸ در ۸ در می‌آید که در ادامه خواهیم دید.

۳-۱-۲ چراغ‌های LED

با توجه به اینکه اتصال چراغ‌های LED تنها نیاز به یک ولتاژ صفر و یک ولتاژ فعال دارد، از توضیح نحوه اتصالشان صرف نظر می‌کنیم. در ادامه می‌توانید کد پیاده‌سازی شده برای روشن یا خاموش کردن چراغ‌ها را مشاهده کنید.

```
import RPi.GPIO as GPIO

class PinHandler:

    # This class, with its methods, helps us to manage LED lights.
    # Actually makes a wrapper to turn on or off the lights
    # easilly with just a method call.

    @staticmethod
    def left_on():
        GPIO.output(Pin.LEFT.value, GPIO.HIGH)

    @staticmethod
    def left_off():
        GPIO.output(Pin.LEFT.value, GPIO.LOW)

    @staticmethod
    def right_on():
        GPIO.output(Pin.RIGHT.value, GPIO.HIGH)

    @staticmethod
    def right_off():
        GPIO.output(Pin.RIGHT.value, GPIO.LOW)
```

همانطور که در کد می‌توان دید، ۲ تابع روشن کردن و ۲ تابع خاموش کردن چراغ داریم که هر جفت از خاموش و روشن کردن‌ها مربوط به یکی از جهات راست یا چپ است.

۳-۱-۳ بخش نرم افزاری اصلی

در کنار کدهای قبلی، یک کد اصلی نیز وجود دارد که مغز متفکر سیستم است و با توجه به شرایط و به تناسب از توابع تعریف شده استفاده می‌کند. در این قسمت بخش‌های مختلف این کد را بررسی می‌کنیم.

در ابتدا کتابخانه‌های مورد نیاز را import می‌کنیم. در اینجا از کتابخانه‌ی logging برای نگه داشتن تاریخچه‌ی تشخیص‌های سیستم از موجودات زنده استفاده می‌کنیم. همانطور که می‌بینید این تاریخچه در فایلی با اسم history.log ذخیره می‌شود.فرمت لاگ خروجی را نیز می‌توانید در زیر ببینید:

2022-12-20 16:07:25,426 - LEFT - 30.0625

سپس تعداد متغیر اولیه تنظیم شده است که هرکدام استفاده خاص خود را دارند. به عنوان مثال متغیرهای MIN_TEMP و MAX_TEMP بازه‌ی دمایی موجودات زنده را برای سیستم مشخص می‌کند.

```
import logging
import numpy as np

MIN_TEMP = 29 # Minimum temperature needed to turn on the lights
MAX_TEMP = 50 # Maximum temperature that can be related to a living thing
```

برای تشخیص موجود زنده دو تابع اصلی وجود دارد. در تابع generate_submatrices پنجره‌ای ۲ در ۲ در نظر گرفته می‌شود و این پنجره روی کل ماتریس ۸ در ۸ حاصل از خواندن داده‌های دوربین حرارتی لغزانده می‌شود و روی هر ۴ درایه از این ماتریس که قرار گرفت، میانگین دماهای این ۴ خانه را محاسبه می‌کند. با توجه به اینکه این میانگین در بازه ی تعریف شده وجود دارد یا نه، تشخیص می‌دهد که موجود زنده جلوی سیستم وجود دارد یا خیر. این میانگین‌گیری برای جلوگیری از خطای احتمالی پیکسل‌های جداگانه است تا سیستم منعطف‌تر کار کند و با کوچیکترین تغییر دما واکنش نشان ندهد. در ادامه می‌توانید کد مربوط به این دو تابع را مشاهده کنید.

```

def generate_submatrices(matrix, sub_size=2):

    # This function generates all possible NxN submatrices of
    # a given matrix.

```

```
    submatrices = []
    for i in range(len(matrix) - sub_size + 1):
        for j in range(len(matrix) - sub_size + 1):
            direction = LEFT

            if j == len(matrix) / 2 - 1:
                direction = MID
            elif j >= len(matrix) / 2:
                direction = RIGHT

            submatrices.append(
                (direction, matrix[i:i + sub_size, j:j + sub_size]))

    return submatrices

def decide_lights(sub_matrices):

    # This function decides which light to turn on
    # based on the location of the high temperature
    # points.

    for direction, sub_matrix in sub_matrices:
        mean = np.mean(sub_matrix)

        if MIN_TEMP <= mean <= MAX_TEMP:
            if direction == LEFT:
                # Turn on the left light
                pin_handler.left_on()
                found_left = True
            elif direction == RIGHT:
                # Turn on the right light
                pin_handler.right_on()
                found_right = True
```

```
            elif direction == MID:
                # Turn on both of them
                pin_handler.left_on()
                pin_handler.right_on()
                found_right = True
                found_left = True

    # If there wasn't any living thing on the left side
    # so, turn off the left light
    if not found_left:
        pin_handler.left_off()

    # If there wasn't any living thing on the right side
    # so, turn off the right light
    if not found_right:
        pin_handler.right_off()
```

در تابع generate_submatrices تمام ماتریس‌های ۲ در ۲ ممکن استخراج می‌شود. همچنین در همین تابع تشخیص داده می‌شود که هرکدام از ماتریس‌های ۲ در ۲ تولید شده، در کدام سمت راننده است، در چپ یا راست. این تشخیص جهت به این دلیل است که چراغ درست روشن شود. اگر موجود زنده در سمت راست بود، چراغ راست و اگر در چپ بود چراغ چپ روشن شود. در تابع decide_lights نیز بر اساس داده‌های تولید شده از تابع قبل، تصمیم گرفته می‌شود که کدام چراغ‌ها روشن و یا خاموش شوند. همچنین فرایند ثبت شناسایی‌ها در تاریخچه نیز در همین تابع انجام می‌گیرد. این کار با صدا زدن تابع log انجام می‌شود. کد مربوط به این تابع را می‌توانید در زیر ببینید.

```

def log(direction, mean, sub_matrix):
    # Save any detection of living thing

    logging.info(f"{direction} - {mean} - {sub_matrix}")
```

در نهایت یک تابع main وجود دارد که در یک حلقه‌ی بینهایت مقادیر دوربین حرارتی را خوانده و توابع تعریف شده در بالا را صدا می‌زند. این کد را می‌توانید در زیر مشاهده کنید:

```
1
2  def main():
3      # This function acts as a coordinator that calls
4      # all needed functions to read data from sensor
5      # and decide to turn on the correct light based
6      # on the position of the detected living thing.
7
8      t0 = time.time()
9      sensor = []
10
11     while (time.time() - t0) < 1:
12         sensor = amg8833_i2c.AMG8833(addr=0x69)
13
14     pixels_resolution = (8, 8)
15     pixels_to_read = 64
16
17     while True:
18         # Read serial data from sensor
19         status, pixels = sensor.read_temp(pixels_to_read)
20         if status:
21             continue
22
23         # The temperature of the thermistor used in the sensor
24         T_thermistor = sensor.read_thermistor()
25
26         # Trasnforming sensor serial data to an 8x8 numpy array
27         pixels_reshaped = np.reshape(pixels, pixels_resolution)
28
29         # Generate all 2x2 submatrices
30         submatrices = generate_submatrices(pixels_reshaped)
31
32         # Turn on/off the lights based on the values of sensor's data
33         decide_lights(submatrices)
```

```
34
35 if __name__ == '__main__':
36     main()
```

در این تابع ابتدا دوربین حرارتی اتصالش برقرار شده سپس همواره ۶۴ پیکسل از آن خوانده می‌شود و به صورت یک ماتریس ۸ در ۸ تبدیل می‌شود. سپس ماتریس‌های ۲ در ۲ تولید شده از آن به تابع decide_lights داده می‌شود تا تصمیمگیری‌های مربوط به چراغ‌ها را انجام دهد. این فرایند تا زمان خرابی سیستم یا قطع آن توسط کاربر انجام خواهد شد.

۳-۱-۴ بسته بندی

این قسمت هنوز طراحی نشده است.

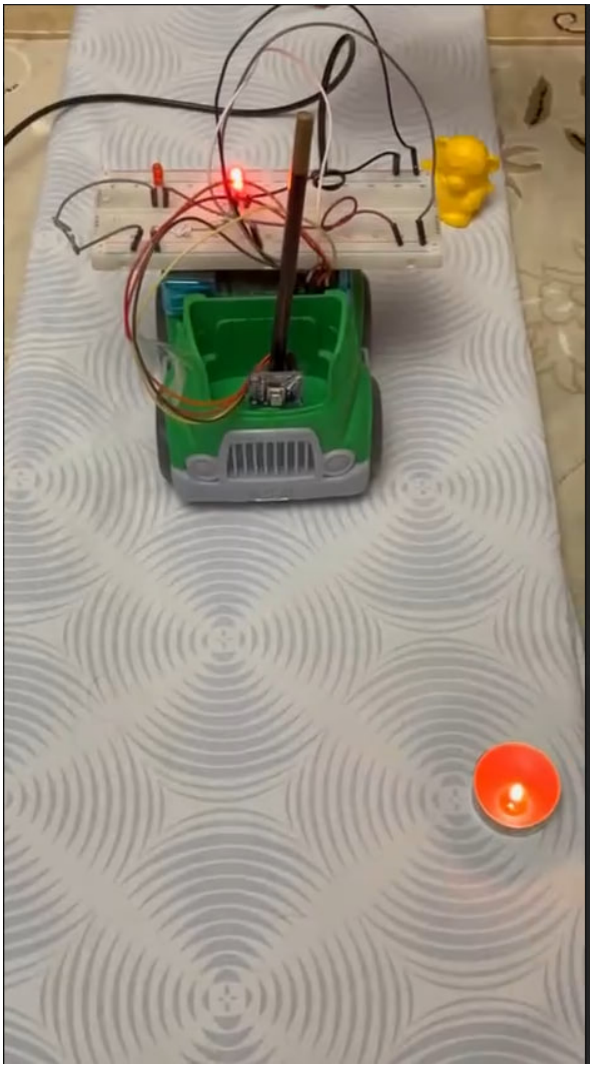
فصل ۴

خروجی

در این قسمت خروجی دستگاه را طی چند اجرای نمونه بررسی می‌کنیم.

۴-۱ تست در روشنایی

در تصویر زیر می‌توانید مشاهده کنید که چرا سمت چپ خودرو در حضور یک منبع حرارتی جلوی سنسور روشن شده است. همچنین این منبع در سمت چپ خودرو قرار دارد.



شکل ۴-۱: تست در روشنایی

در زیر می‌توانید مقداری که اجرای کد، در خروجی استاندار (stdout) خود نمایش می‌دهد قبل و بعد از روشن شدن چراغ مشاهده کنید.

آرایه خروجی زیر، قبل از قرار گرفتن شمع در برابر خودرو است. همانطور که می‌بینید تمام دماهای نشان داده شده برای هر پیکسل، تقریبا برابر همان دمای محیط است.

```
[[23.1 24.4 24.4 23.2 24.7 23.6 23.8 23.8]
 [24.1 24.4 24.9 23.7 24.8 23.5 23.9 23.3]
 [23.6 24.  24.7 24.3 24.7 24.9 24.7 24. ]
 [24.9 23.2 23.7 24.9 24.3 23.6 23.9 23.7]
 [23.3 24.  23.  24.9 24.8 24.2 23.7 23.9]
 [23.8 23.6 23.  23.3 24.9 24.1 24.6 23.6]
```

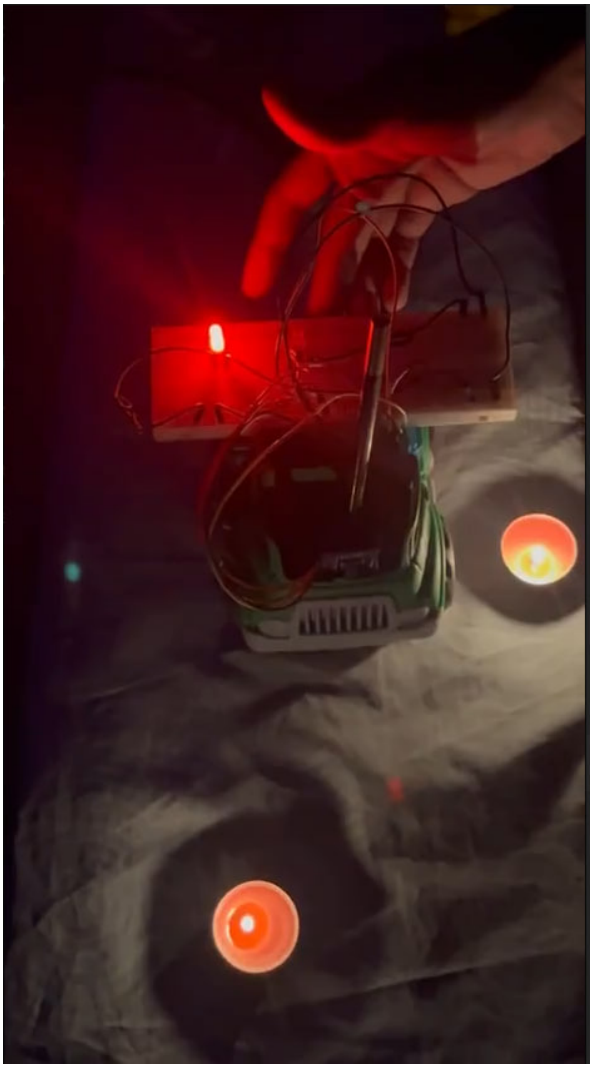
```
 [23.3 23.5 24.8 23.3 23.3 24.2 24.  24.9]
 [23.  23.2 24.7 23.  24.3 23.9 23.  24.6]]
```

اما پس از قرار گیری منبع گرما در برابر خودرو، تغییری ملموس در سمت راست ماتریس خروجی مشاهده می‌شود که نشان از افزایش دما در ناحیه خاصی از محیط دارد. این تغییر دما به احتما قوی می‌تواند مربوط به حضور یک موجود زنده در جلوی خودرو باشد. پس در همین زمان است که دستگاه با روشن کردن چراغ سمت چپ، به راننده هشدار می‌دهد که موجودی زنده در سمت چپ او قرار دارد.

```
[[23.8 24.8 24.2 24.4 23.4 39.6 40.4 40.2]
 [23.3 23.4 24.4 23.4 23.5 40.4 39.9 39.9]
 [24.6 23.8 23.6 24.2 24.8 39.2 40.4 39. ]
 [23.  24.2 24.9 24.5 23.  47.3 39.2 49.8]
 [23.5 24.  24.1 23.8 24.5 49.9 40.2 48.5]
 [23.4 24.2 24.4 24.4 23.4 39.  40.9 39.3]
 [23.6 23.6 23.7 24.2 23.2 40.6 39.9 39.5]
 [23.5 24.4 23.5 24.8 23.1 40.3 39.7 40.1]]
```

۴-۲ تست در تاریکی

در تصویر زیر می‌توانید مشاهده کنید که چرا سمت راست خودرو در حضور یک منبع حرارتی جلوی سنسور روشن شده است. همچنین این منبع در سمت راست خودرو قرار دارد.



شکل ۴-۲: تست در تاریکی

در زیر می‌توانید مقداری که اجرای کد، در خروجی استاندار (stdout) خود نمایش می‌دهد قبل و بعد از روشن شدن چراغ مشاهده کنید.

آرایه خروجی زیر، قبل از قرار گرفتن شمع در برابر خودرو است. همانطور که می‌بینید تمام دماهای نشان داده شده برای هر پیکسل، تقریبا برابر همان دمای محیط است.

```
[[23.1 23.  23.7 24.3 24.1 24.7 24.2 23.4]
 [24.2 23.4 24.  24.5 24.7 23.1 23.9 24.9]
 [24.4 23.3 24.9 24.9 24.4 24.9 23.6 23.1]
 [24.4 23.7 23.4 24.7 23.9 24.3 23.6 23.3]
 [24.4 24.3 23.3 23.1 24.2 23.9 23.5 23.4]
 [23.5 23.2 24.8 23.8 23.5 23.7 23.7 24.1]
```

```
 [23.6 23.7 23.5 24.5 23.1 23.8 24.6 23. ]
 [24.1 24.2 24.2 24.4 23.2 24.1 24.4 24. ]]
```

اما پس از قرار گیری منبع گرما در برابر خودرو، تغییری ملموس در سمت چپ ماتریس خروجی مشاهده می‌شود که نشان از افزایش دما در ناحیه خاصی از محیط دارد. این تغییر دما به احتما قوی می‌تواند مربوط به حضور یک موجود زنده در جلوی خودرو باشد. پس در همین زمان است که دستگاه با روشن کردن چراغ سمت راست، به راننده هشدار می‌دهد که موجودی زنده در سمت راست او قرار دارد.

```
[[45.1 45.  45.7 46.3 24.1 24.7 24.2 23.4]
 [46.2 45.4 46.  46.5 24.7 23.1 23.9 24.9]
 [46.4 45.3 46.9 46.9 24.4 24.9 23.6 23.1]
 [51.4 50.7 45.4 46.7 23.9 24.3 23.6 23.3]
 [54.4 53.3 45.3 45.1 24.2 23.9 23.5 23.4]
 [45.5 45.2 46.8 45.8 23.5 23.7 23.7 24.1]
 [45.6 45.7 45.5 46.5 23.1 23.8 24.6 23. ]
 [46.1 46.2 46.2 46.4 23.2 24.1 24.4 24. ]]
```

فصل ۵

قیمت

یکی از مسائل مهم در طراحی محصول قیمت آن است. البته با توجه به این که این محصول به صورت نمونه اولیه طراحی شده است، طبیعتا قیمت تمام شده آن از محصولی که بخواهد تولید عمده بشود بالاتر خواهد بود. در جدول ۵-۱ قیمتی تخمین زده شده و هزینه نهایی پروژه آورده شده است.

ردیف	قطعه	قیمت تخمینی	قیمت نهایی
۱	Raspberry PI 3B	۳۱۰۰	۰
۲	AMG8833	۱۱۰۰	۱۱۰۰
۳	Flash USB	۱۰۰	۱۳۰
۴	LED	۲۵	۰
۵	Board	۲۵	۰
۱۸	Total	۴۳۵۰	۱۲۳۰

جدول ۵-۱: جدول قیمت محصول (قیمت‌ها به واحد هزارتومان)

فصل ۶

جمع‌بندی

در این پروژه به پیاده‌سازی سیستم دید در شب اتومبیل پرداختیم که به راننده برای جلوگیری از سانحه در محیط‌های تاریک کمک می‌کند. در این سیستم با استفاده از سنسور دوربین حرارتی که از طریق جذب مادون قرمز عمل می‌کند و همچنین واحد پردازشی رزپری‌پای، حضور یک موجود زنده جلوی دید راننده را تشخیص و به کمک چراغ‌ها به اون هشدار دادیم.

در کنار طراحی کلی و نحوه پیاده‌سازی، تمام کدهای مربوط به این محصول نیز به صورت متن باز در گیتهاب پروژه قراره گرفته است و هرکسی می‌تواند با کمک این کدها به بهبود و ارتقاء این سیستم کمک کند و یا با الهام از آن، پیاده سازی خاص خودش را ارائه دهد.

آنچه محصول ما را از دیگر محصول‌ها متمایز می‌کند قیمت بسیار پایین تمام‌شده‌ی آن است که می‌توان حتی بیشتر آن را کاهش داد. به عنوان مثال با استفاده از آردوینو به جای رزپری و همچنین استفاده از چراغ‌های خودرو به جای چراغ‌های مجزا برای سیستم. حتی می‌توان در هنگام پیاده‌سازی این سیستم برای خودرو، کد آن را به عنوان یک نرم افزار در اختیار کامپیوتر خودرو قرار داد و تمام مسئولیت‌های رزپری را به کامپیوتر اتومبیل سپرد.