PARKOLITE Ltd

ParkoLite Ltd.
6189 Lochdale Street
Burnaby, British Columbia
Canada V5B 2M6

Dr. Andrew Rawicz
School of Engineering Science
Applied Sciences Building
8888 University Drive
Burnaby, British Columbia
Canada V5A 1S6

November 9, 2015

Re: ENSC 440W Design Specification for Alternative Parking Indicator

Dear Dr. Andrew Rawicz,

Attached is a document describing the Design specification for *Alternative Parking Indicator* (API).We at ParkoLite Ltd. aim to design and implement a parking indicator that will make finding a parking spot an easier task for drivers. Our design consists of a pole with RGBY LEDs and electronic circuitry that will display the availability of a parking spot to a driver.

The purpose of this document is to outline the detail description of electrical, software and mechanical requirements along with that this document will also highlight the methods and procedures that will be used to test the different components of API.

ParkoLite consists of 5 talented and dedicated engineers: Raj Sidhu, Soudeh Mousavi, Mubaarak Sandhu, Oliver Krajci, and Azin Navah. If you have any questions regarding our Design Specification or product, please feel free to contact us at gssidhu@sfu.ca or by phone at (778) 239-4676.

Sincerely,

Raj Sidhu

[illegible]

CEO
ParkoLite Ltd. Enclosure: Design Specification for Alternative Parking Indicator

ParkoLite Design Specification

Design Specification for Alternative Parking Indicator

ENSC305W/440W: Capstone Engineering Science Project

Project Team:

Raj Sidhu

Soudeh Mousavi

Mubaarak Sandhu

Oliver Krajci

Azin Navah

Submitted to:

Dr. Andrew Rawics ENSC440W

Professor Steve Whitmore ENSC305W

Faculty of Applied Sciences

Simon Fraser University

Contact Person:

Raj Sidhu

gssidhu@sfu.ca

Issue Date:

November 9, 2015

Revision Number: 1

Abstract

The document contains the detailed design specifications of the *Alternative Parking Indicator* (API) developed by Parkolite Ltd. Our team's goal is to successfully create and implement a parking indicator which will provide assistance for drivers to look for a parking spot and also help the parking enforcement officers to issue tickets to violators. This system is a pole design based model and will be installed on top the parking curb.

In this design document, we will give detailed design descriptions of electrical, software and mechanical components that are required for the successful implementation of the API system. API will consist of AMR sensors, RGB SMD 5050 LEDs and electronic circuitry, which will all be embedded into a very well designed pole model.

Design specification will provide details on the test plans that will be carried out throughout the design and implementation process to ensure the success of our project. This document will also illustrate the communication between software and various electrical components and is designed for proof of concept model.

Table of Contents

LIST OF TABLES V

TABLE OF FIGURES V

GLOSSARY VI

1. INTRODUCTION 1
1.1 Scope 2
1.2 Intended Audience 2
1.3 Background 2

2. SYSTEM OVERVIEW 3

3. SENSORS 5
3.1 Sensor Placement 6
3.2 Sensor Circuitry 7

4. LEDS 8
4.1 Characteristics 8
4.2. Circuitry and Electrical Connections 9

5. MICROCONTROLLER UNIT 10

6. MECHANICAL POLE DESIGN 11
6.1 Product Design for Prototype 11
6.2 Proof of Concept Design 13

7. DIGITAL POTENTIOMETER 15
7.1 Digital Potentiometer Pin descriptions 16
7.2 Digital Potentiometer Programming 16

8. PCB/CIRCUIT DESIGN 17
8.1 Sensor and supporting circuitry 17
8.2 Microcontroller and Wi-Fi module 19
8.3 Power elements 19
8.4 PCB Design 19

9. SOFTWARE/APP 20

10. USER INTERFACE 21

11. TEST PLANS 22
11.1 Unit Acceptance Testing – UAT Phase 22
11.1.1 Microcontroller Unit Test Plan 22
11.1.2 Digital Potentiometer Unit Test Plan 22
11.1.3 AMR Sensor Unit Test Plan 22
11.1.4 LED Unit Test Plan 22
11.2 Integration Testing Phase 22

11.2.1AMR Sensor Integration Test Plan *22*
11.2.2 Potentiometer Integration Test Plan *22*
11.2.3 Microcontroller integration Test Plan *23*
11.2.4 LEDs integration Test Plan *23*
11.3 REGRESSION TESTING PHASE 23
11.4 APP INTEGRATION PHASE TESTING 23
12. CONCLUDING REMARKS 24
REFERENCES 25
APPENDIX – CONTROLLER PROGRAMMING UNFINISHED 26

List of Tables

Table 1 - Flux Density Shift table 6
Table 2 - Characteristics of SMD505 LED strip 8
Table 3 - LED circuitry part specifications 9
Table 4 - Arduino Specifications 10
Table 5 - Materials, parts and dimensions 12
Table 6 - Material specifications for tungsten carbide 12
Table 7 - Steel 300W refers to wieldable steel 13
Table 8 - High Density polyethylene material characteristics 14
Table 9 - Control Inputs Pin Description 16

Table of Figures

Figure 1 - Model overview 1
Figure 2 - Original design put forth in Proposal 3
Figure 3 - New Proposed design for API 4
Figure 4 - AMR Sensor Bridge 5
Figure 5 - X - axis Direction Sensing 6
Figure 6 - Singe Axis Detection Circuit for AMR 7
Figure 7 - Alternative parking indicator states controlled by arduino 8
Figure 8 - API microcontroller - LED circuitry 9
Figure 9 - Arduino MCU 10
Figure 10 - metal base with internal core including flanking rods and core rod 11
Figure 11 - metal cap that allows the metal poles to slot in and fasten into place 12
Figure 12 - High density polyethylene outer shell of API 14
Figure 13 - Proof-of-concept mechanical prototype 15
Figure 14 - Digital Potentiometer top view 15
Figure 15 - Wiper Mode Selection 16
Figure 16 - AC Timing Diagram Showing Wiper Dependence on Control Inputs 17
Figure 17 - Charge Pump for Degaussing the AMR Sensor 18
Figure 18 - PCB Design for All Ci 20
Figure 19 - Drivers Perspective 21

Glossary

API	Alternative Parking Indicator
AMR	Anisotropic magneto-resistance
CSA	Canadian Standards Association
EPG	Engineering Policy Guide
LED	Light-emitting Diode
MCU	Microcontroller Unit
RGBY	Red, Green, Blue, Yellow

1. Introduction

At ParkoLite we are designing an Alternative Parking Indicator (API) device (Figure 1), a cylindrical two feet high pole that has a translucent body for mounting LEDs; API is to be installed on the streets of Vancouver and lower mainland where there are bylaws to be followed by drivers to park their vehicles. API aids drivers to find an available parking spot faster by reducing the time spent on figuring out the street signs. Not only the API helps the mass public but it also helps business and parking attendants throughout the city. Using AMR sensors, API sends signals to Arduino -ATmega 328P chip. These signals are then processed and the appropriate output LED colour appears across the non-interactive user interface with regards to the pole itself. This document lays out the design specifications of the API device and potential future design considerations such as developing a mobile phone application.

This document unlike the functional specifications outlines the specific design details. There was a systematic process in place while designing this API. The design specifications below will go through a detailed approach, which lead Parkolite ltd to the prototype seen in Figure 1.

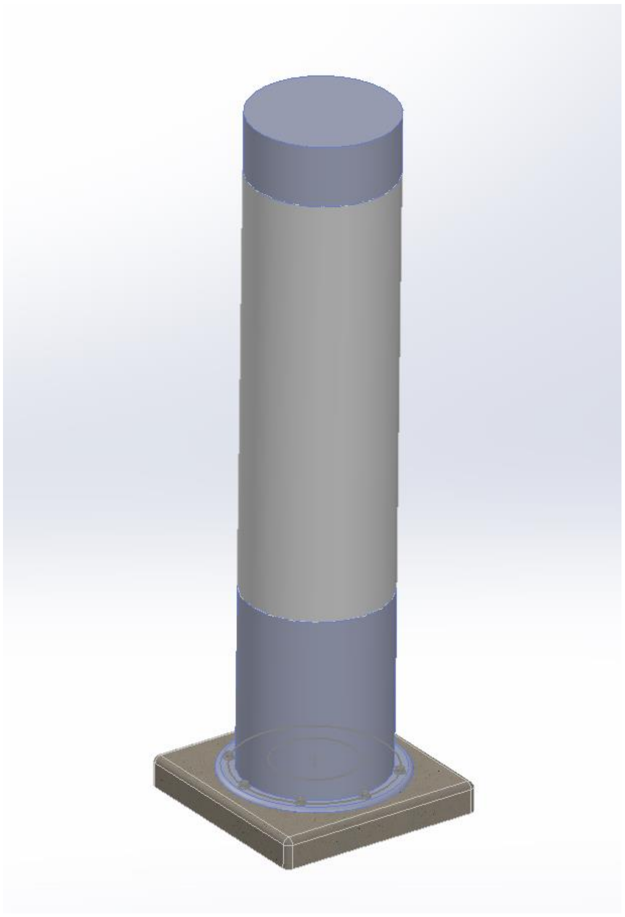


Figure 1 – Model overview

1.1 Scope

The scope of this document is to outline the design requirements of the Alternative Parking Indicator device developed by ParkoLite Ltd. and supports Functional Specification for API document. This document describes in detail the layouts of the electrical, mechanical and firmware design of the system, which includes all electrical, mechanical and firmware requirements for the proof of concept model; the details of these parts will be included in their specific overview throughout the document. This document also proposes a system test plan for API system. We will be discussing the changes we had in our mechanical design put forth in our *Proposal* document of the overall system.

1.2 Intended Audience

The Design specification is intended to be used by all members at ParkoLite. The team members will refer to this document in every step of development to ensure that the final product meets the predefined functional requirements and design specifications. This document is to be used for references in testing phase of the design; this document is also used for educational purposes for the future stake holders, investors and potential pilot programs with certain municipalities.

1.3 Background

The following statistic shows Vancouver Parking Fines/ Towing Statistics; please note these statistics are selected from the proposal document. (City of Vancouver, 2012)

- Vancouver collected 25 million dollars from parking fines (if wealth is spread evenly amongst all Vancouver residents it would equal $41.43 per person in the city of Vancouver)
- Currently City of Vancouver has 95% conviction rate for parking fines 80% province wide
- City will tow illegally parked vehicles if they have three unpaid tickets over past two years
- According to the new policy cars will be towed if offenders have had more than 5 tickets within the past two years, even if they had paid out their tickets

- City of Vancouver paid Busters Towing, $754,910.00 in 2012
- It is estimated that 25,000 vehicles are towed each year at an average cost of $150.00 per tow = $3,750,000.00 dollars
- In Q2 of 2015 Vancouver had an actual revenue from parking of 15.9 million dollars with an annual forecast from parking revenue being approximately 53.2 million dollars (City of Vancouver, 2015)

A fraction of parking penalties is the consequence of misleading and confusing parking signs – bylaws. Indeed, drivers need to be more careful reading the signs on the streets to avoid the parking tickets or/ and the towing expenses.

We, at ParkoLite believe that we are bringing a solution to reduce these penalties and confusions for drivers; we also believe that the parking violators must be ticketed. API device provides an effective visual feedback, both for drivers, public and for the city parking enforcement officers. For the Officer where they can easily spot the violators and ticket them; for drivers where they will be able to confidently park their vehicle knowing they will not be towed.

2. System Overview

In our initial design, we placed the API flush with the ground as it seemed like the best way forward. However, after careful analysis and discussing and studying the benefits versus the potential pitfalls, it was decided that the best approach would be to use an erect API's that would be 90° to the surface.

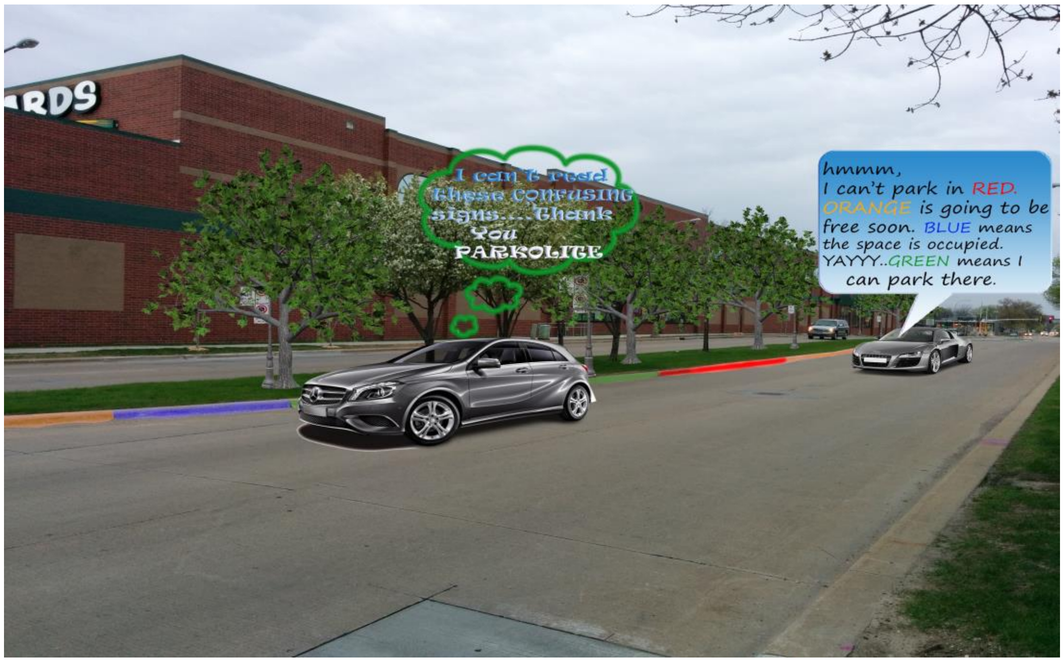

RDS
I can't read these confusing signs....Thank You PARKOLITE
hmmm, I can't park in RED. ORANGE is going to be free soon. BLUE means the space is occupied. YAYYY..GREEN means I can park there.

Figure 2 - Original design put forth in Proposal

The benefits of using this new design can be seen below. From many aspects this new design not only takes up less space, it can also be integrated into the existing electrical infrastructure more smoothly.

- The standing API offers much better visibility to the drivers compared to the curb side API.
    - The meter's height let the drivers see the API better and more clearly.
    - The API offers much better visibility in bad weather conditions, especially when it is raining or snowing. Central Canada is known to have over 2ft of snow during the winter seasons at any time depending on the year.
    - It is less likely for the API to become unreadable due to it becoming dirty and dusty.
- The standing Parkolite is less likely to become damaged and is easy to repair due to its modular design.
    - The curb side Parkolite is more likely to be damaged by parking cars and in case it is damaged, the entire LED panel needs to be replaced.
- The standing Parkolite is much more cost efficient than the curb side meter.
    - The curb side meter contains more LED lights as well as more water-proofed plastic coverings for the lights.

All though the design seen below is not to scale, it gives the reader a rough idea as to what Parkolite has decided to move forward with.

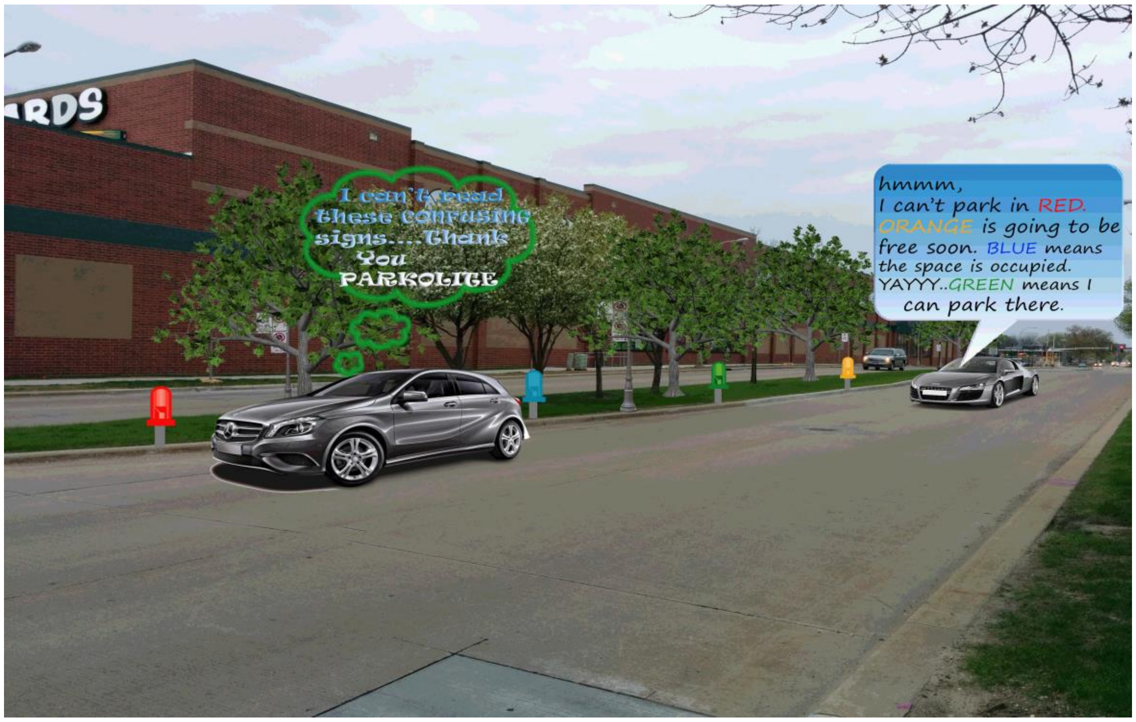

I can't read these confusing signs....Thank You PARKOLITE
hmmm, I can't park in RED. ORANGE is going to be free soon. BLUE means the space is occupied. YAYYY..GREEN means I can park there.

Figure 3 - New Proposed design for API

As the driver comes to this new indicator he/she will find that the light is either green or red. If the light is green then the driver has the ability to park in that spot. However if he light indicates that it is red then the driver should avoid parking in that spot.

Once the driver has parked, after a short delay the light will turn blue indicating that he parking spot has been occupied. Depending on the bylaws in the area the driver may be able to park for 30 minutes or three hours. Based on the area the light will change from blue to yellow indicating that the driver has only 15 minutes remaining for parking. With the integration of a mobile app the driver will be able to know whether or not the car should be moved.

Once the light has changed from yellow to red it indicates any parking enforcement operators that the vehicle is in direct violation of the bylaws. The driver now risks being ticketed for over use of the parking spot. However, if the driver leaves before the parking spot turns red or even after it turns red it will turn green indicating that it has been vacated. This is the basic system overview as to how the API will work in the public setting. The mobile application is in the development stage and may not be ready for the proof-of-concept stage. However, as the prototype is implemented it will be a part of the design.

3. Sensors

The sensors used by ParkoLite for the detection of vehicles are AMR sensors, which has all the resistive elements oriented as a "Wheatstone bridge" configuration as shown in the figure below. AMR sensors are used because they do not have orientation issues like other sensors such as ultrasound or infrared. Furthermore, they are very cost effective with regards to the sensors market.

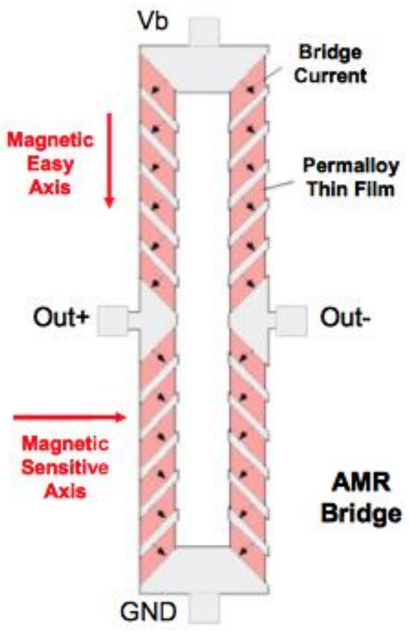


Figure 4 - AMR Sensor Bridge

The resistance of the resistive element changes as magnetic field upon each element changes and each bridge consists of four resistive elements with opposite elements being similar. Total amount of the output voltage for AMR senor is measured from Out+ to Out- as shown in the following equation.

$$\begin{aligned}&Out+ - Out- = S * Vb * Bs \; with,\\ &S = Sensitivity \; (nominally \; 1mV/V/gauss)\\ &Vb = Bridge \; Supply \; Voltage \; in \; volts\\ &Bs = Bridge \; Applied \; Magnetic \; Flux \; in \; gauss\end{aligned}$$

3.1 Sensor Placement

AMR sensors will detect the change in the magnetic field when a vehicle approaches the sensor. For this initial proof-of-concept stage Parkolite is using the single direction sensing. In which X-Axis direction is the direction in which the sensing occurs. The following diagram shows the vehicle direction sensing.

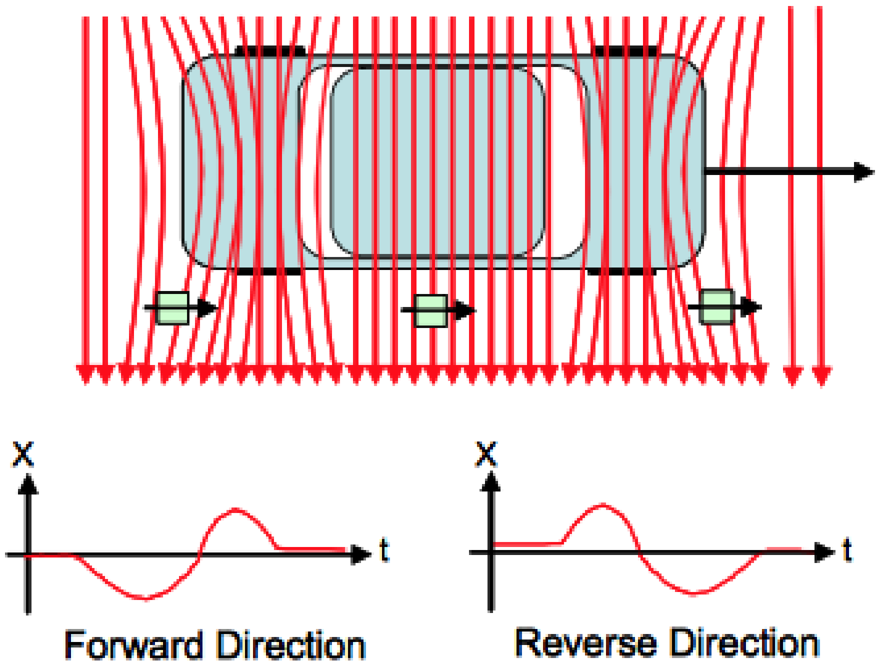


Figure 5 - X - axis Direction Sensing

As seen from the figure above that by placing the sensor on the sides or at some height (In our project it will be placed at top section of the pole design), vehicle detection can be done. API is using X-axis direction sensing. Lines of magnetic flux will bend at the sensor towards the vehicle as vehicle approaches the sensor, which will decrease the flux density, more detailed description of the values of flux density are shown in the table below

Table 1 - Flux Density Shift table

Standoff Distance Vs. Flux Density	
Standoff Distance (in feet)	Flux Density Shift (in milligauss)
1	**270**
3	**75**
5	**10**
10	**2**
12	**<1**

As flux density decreases, signature voltage from sensor will be negative. When the vehicle vacates the parking spot, flux density will change again and signature voltage from sensor will be positive. However if vehicle backs up and parks again, senor's efficiency will not be hindered and it will just do the same thing but signature voltage plot will look like a mirror image as shown in the figure 5.

3.2 Sensor Circuitry

A simple analog circuit of single-axis AMR sensing is shown in the following diagram, selecting HMC1021S for simple prototyping purposes.

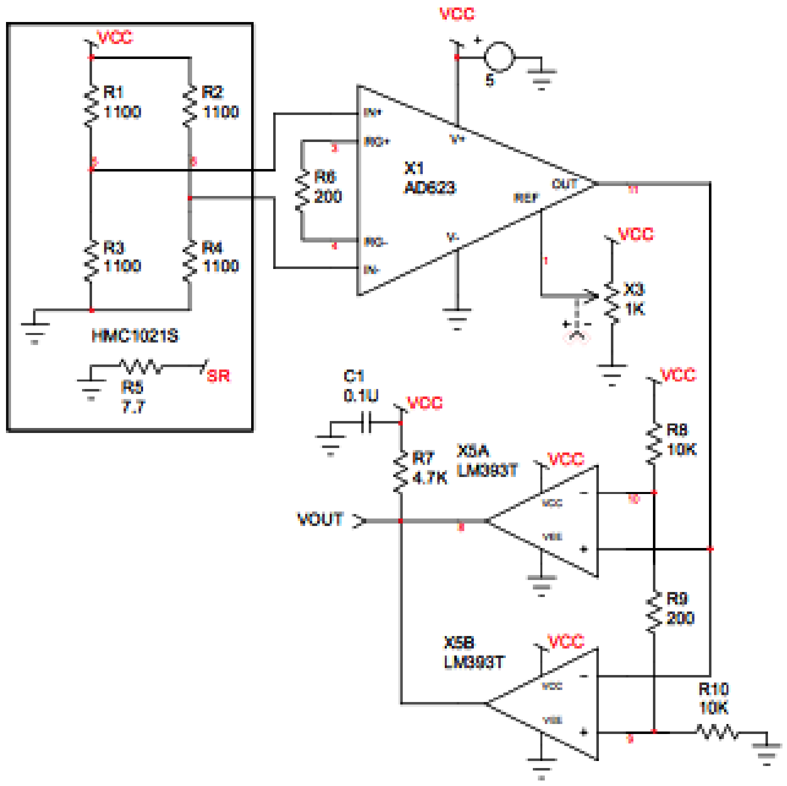


Figure 6 - Singe Axis Detection Circuit for AMR

The basic necessity of the single axis AMR sensor for the project is to get close to 2.5V value for the amplifier output. Hence resistors R8, R9 and R10 are chosen if output is significantly off from 2.5V one of the comparators will put the output to logic low state. Vehicle detection distance can be adjusted by changing/adjusting the value of R9.

The circuitry is susceptible to false scenarios due to earth's magnetic field for which we tried to re-trim the potentiometer to re-center to 2.5V bias value when no vehicle is present but this still gave us some issues. So the alternative and best solution is to use a digital potentiometer to fix this issue. All the functionality of the digital potentiometer is shown later in this document in electrical design stage.

4. LEDs

The LED lights notify drivers and parking enforcement officer what the status of the parking spot is. In other words, it provides visual feedback to drivers by changing LED colors depending on what the firmware executes at that specific time. Figure 7, shows API device once is not powered.

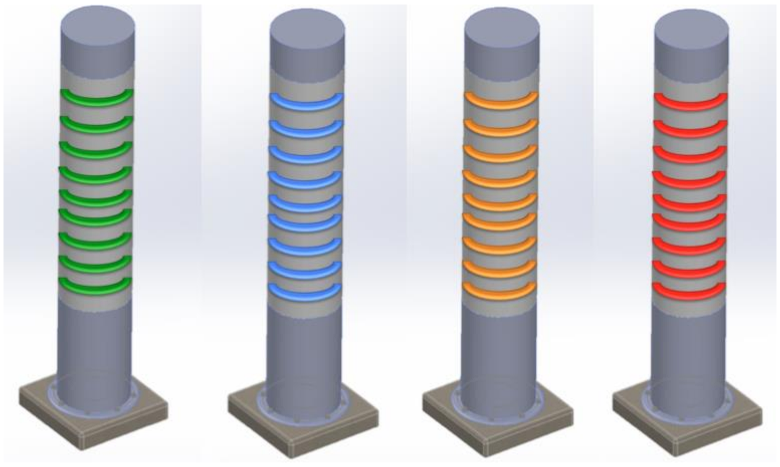


Figure 7 - Alternative parking indicator states controlled by arduino

Note that we have changed the type of LED we had used in the Functional Specification, the general requirements for SMD 5050 RGB LED Strip is the same as SMD 5050 RGB LED Strips mentioned in Section 3.1 of Functional Specification. SMD 5050 RGB LEDs strip with double-sided tape makes it easier to mount; these LEDs' share a common anode and three distinct cathodes for each one of the three colours.

4.1 Characteristics

Table 2 - Characteristics of SMD5050 LED strip

SMD 5050 LED wave/luminosity characteristics			
Emitting color	Wavelength (nm)	Luminous Flux (Lm)	LED Work Current (mA)
RED	**625 - 630**	**90 – 120**	**170 – 190**
GREEN	**515 – 520**	**90 – 120**	**170 – 190**
BLUE	**465 - 570**	**80 – 90**	**170 – 190**

4.2. Circuitry and Electrical Connections

SMD 5050 is individually controllable, therefore, a microcontroller is needed To meet the LED requirements mentioned in [R28-PII] in Functional Specification, Arduino is programmed to control LED's and outputs the expected result in terms of color, Figure 3 shows SMD 5050 LED Strips laminating once the Arduino receives an interrupt and send a signal to LED circuitry to output a correct color. As shown in Figure 3, we are using the very similar colors to standard traffic pattern as it mentioned in [R3-PII].

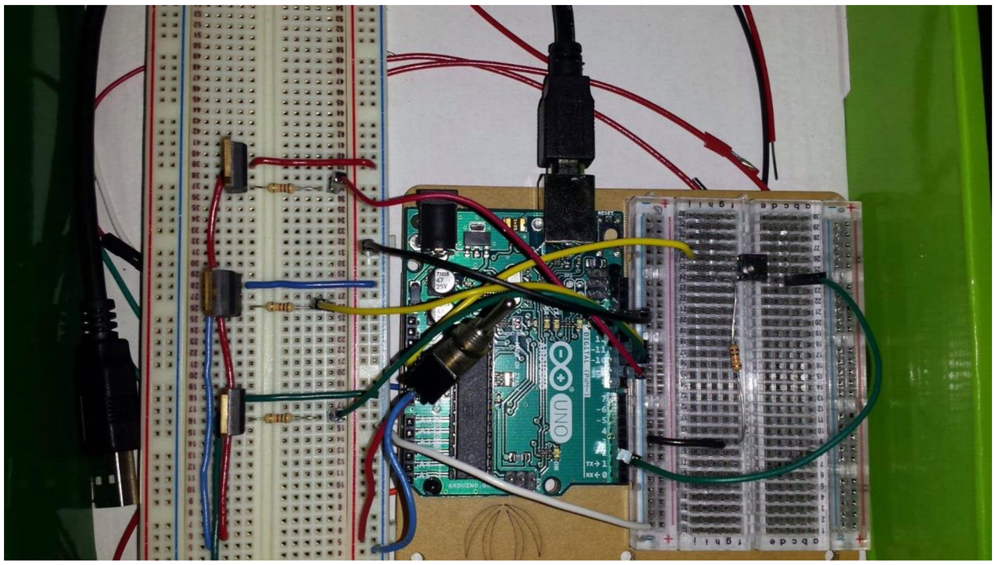


Figure 8 - API microcontroller - LED circuitry

The circuit seen in figure 8 in conjunction with the code in the appendix is responsible for working with the AMR sensor and digital potentiometer bring the whole system together. The circuit above is composed of the following parts which can be seen in table 3.

Table 3 - LED circuitry part specifications

API LED control circuitry	
Parts	Specifications
TIP41C high power BJT (Red)	$I_c = 6A$, $I_B = 3A$, $V_{EB} = 5V$, $P_{TOT} = 65W$ (Max ratings)
TIP41C high power BJT (Blue)	$I_c = 6A$, $I_B = 3A$, $V_{EB} = 5V$, $P_{TOT} = 65W$ (Max ratings)
TIP41C high power BJT (green)	$I_c = 6A$, $I_B = 3A$, $V_{EB} = 5V$, $P_{TOT} = 65W$ (Max ratings)
3 x 1KΩ resistors	N/A
1 x push button to simulate AMR Sensor interrupt	N/A
1x 10KΩ Resistor	N/A
Power switch to turn power on and off for coding	3A 250 VAC or 6A 125 VAC
12V power supply to	In: 100-240VAC, 1.5A; Out: 12V, 6A

5. Microcontroller Unit

To program the digital potentiometer, LEDs to display desired results, free bylaws and other tasks of our project we used arduino UNO. We picked arduino UNO because prototyping and debugging can be done very efficiently.

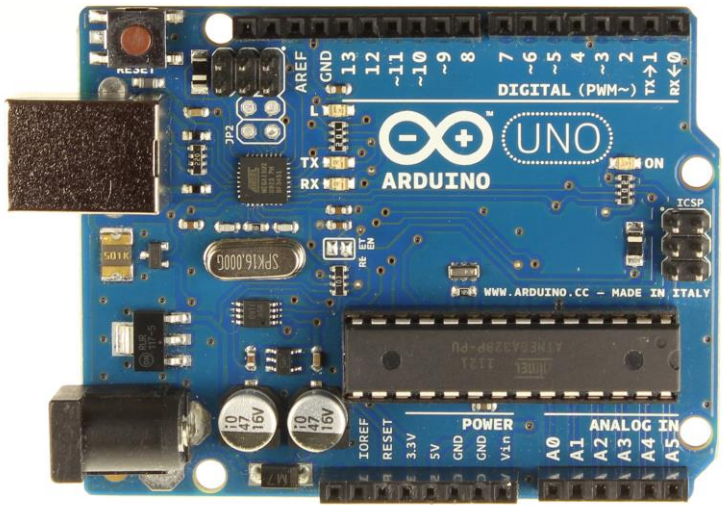


Figure 9 - Arduino MCU

We used the above shown Arduino uno MCU and all of the programming was done in C/C++ environment. Arduino UNO has following technical specifications

Table 4 - Arduino Specifications

Arduino control specifications and usage	
Parameter	**Specification**
Microcontroller	ATmega328
Operating Voltage	5V
Clock Speed	16 MHz
SRAM	2Kbytes
Total digital I/O Ports	14
Analog Input Pins	6
Flash Memory	32 Kb of which 0.5 kb is used by bootloader

This is a good design practice because UNO offers large number of ports and has 2 KB of RAM, hence there will be no shortage of memory and arduino can perform the desired tasks and successfully drive other devices.

6. Mechanical Pole Design

After careful consideration and taking the design to the drawing board. The initial design that was put forth in proposal had considerable issues when it came to visibility and impact factors. By putting the indicator flush with the curb Parkolite weighed the benefits and cost factors of potentially risking the indicator being crushed by reckless vehicle operation. Not only that, if the indicator was to be installed in areas where snow and hail are known to accumulate on side streets the initial proposed design was not a viable option.

6.1 Product Design for Prototype

A new mechanical design was proposed and adopted. The materials were selected to give optimum visibility and maximum strength. For the proof-of-concept/prototype all the material is plastic based. Once Parkolite has decided to proceed to production the design will be a combination of both metal and plastic. As it can be seen in figure 1, the design is approximately 60% metal non-ferrous and 40% translucent polyethylene (HDPE). The following parts will be metal:

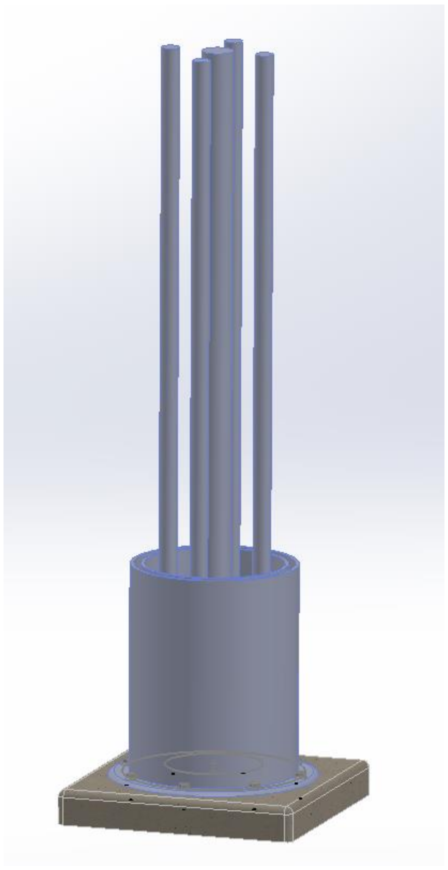


Figure 10 - metal base with internal core including flanking rods and core rod

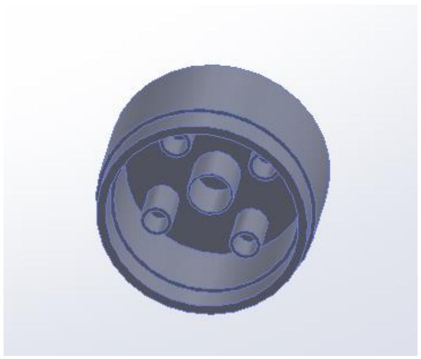


Figure 11 - metal cap that allows the metal poles to slot in and fasten into place

Both parts seen in figure 10 & 11 will be made of the steel. This steel will be the same material that street light poles are made up of. The following table will summarize the strength characteristics along with the dimensions of both parts in figure 10 & 11.

Table 5 - Materials, parts and dimensions

Mechanical parts and material grades			
Part	**ID**	**OD**	**Material**
Outer Shell	10.00cm	12.00cm	High density polyethylene translucent/white
Base	10.00cm	12.00cm	Galvanized(zinc coating) Steel 300W (G45) CSA-G40.21
Cap	10.00cm	12.00cm	Galvanized(zinc coating) Steel 300W (G45) CSA-G40.21
Flank rod threaded 1	0.00cm	1.27cm	Tungsten carbide rod non magnetic
Flank rod threaded 2	0.00cm	1.27cm	Tungsten carbide rod non magnetic
Flank rod threaded 3	0.00cm	1.27cm	Tungsten carbide rod non magnetic
Flank rod threaded 4	0.00cm	1.27cm	Tungsten carbide rod non magnetic
Center rod threaded	0.00cm	2.225cm	Tungsten carbide rod non magnetic

Table 6 - Material specifications for tungsten carbide

Tungsten Carbide		
Property	Minimum Value	Maximum Value
Density (Mg/m^3)	15.25	15.88
Compressive Strength (MPa)	3347	6833
Ductility	0.005	0.0074
Hardness (MPa)	17000	36000
Shear Modulus (GPa)	243	283
Tensile Strength (MPa)	370	530
Service temperature (K)	0	3193
Specific Heat Capacity (J/kg.k)	184	292
Thermal Expansion (10^{-6}/k)	4.5	7.1
Resistivity (10$^{-8}\Omega$.m)	41.7	100

Based on table tungsten carbide was the ideal material for the internal core due to its lack of magnetic properties. Our electrical equipment is heavily based on the change in magnetic fields there for having a material that was magnetic in nature was out of the question. Not only is the material not magnetic it is flame resistant, resistant to oxidation, resistant to corrosion by strong acids and alkalis. Since the application is for outdoor uses it is also extremely resistive to ultraviolet radiation. (AZO materials , 2015)

Table 7 - Steel 300W refers to wieldable steel

Carbon Steel 300W	
Property	Value
Yield Strength (MPa) F_y	300
Tensile Strength (MPa)F_u	450
Density kg/m^3	7,850
Unit Weight (kN/m^3)	77
Modulus of Elasticity(MPa)	200,000
Shear Modulus (MPa)	77,000
Thermal Coefficient(10^{-6}/°C)	11.7
Poisson Ration	0.3

With information seen in table 7 (Web Civil, 2015) it can be noted that it is ideal for our application. Grade 300W is typically susceptible to oxidation in the elements. However by providing a zinc coating the material become galvanizing, which negates the oxidization process thereby making it withstand the elements. Although carbon steel is slightly magnetic in nature by coating it with zinc its magnetic properties become drastically reduced. Not only that by making sure that the steel only exists at the top of the pole and the bottom of the pole it acts a natural shield for unwanted changes in magnetic fields that would disrupt our sensors.

6.2 Proof of Concept Design

For the proof of concept however the entire model is made up of PVC cylindrical piping and fittings with an acrylic cylindrical tube encapsulating the internal components. The reason for choosing this more simplified design for the proof of concept stage was to show that the the concept is a viable solution to an every day issue. This design was specifically chosen because of cost and time factors. Fabricating the design above would have taken weeks to a couple months. Not only that a customized design would require a significant capital investment upfront. By using readily avaiable material that can be found in any hardware store a simple proof-of-concept protoype has been created entirely out of plastic. It does not have the same structural, strength and environmental resistive components that he production ready mechanical design has. The proof-of-concept protoype can be seen below in figure 13. The electronics, LEDs and batteries have yet to be added to the mechanical design. The design in figure 13 is the bare bones of the proof-of-concept prototype.

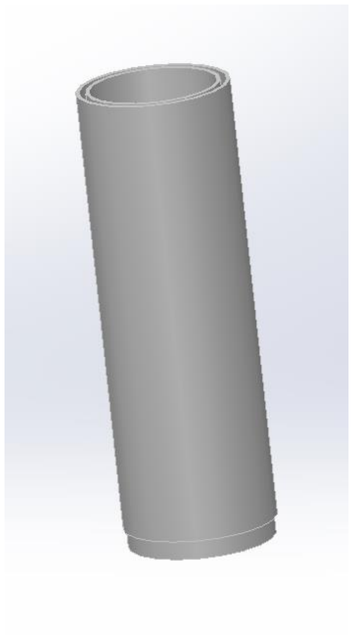


Figure 12 - High density polyethylene outer shell of API

The final component to the design is a high density polyethylene outer shell. This outer shell provides not only structural support but also environmental protection from acid rain, to extreme cold and extreme heat. HDPE is an extremely resilient material and its applications are vast. From transporting oil through pipelines to being integrated in everyday consumer products. HDPE was the ideal choice, for this prototype because it is cheap lightweight and extremely durable. The following table summarizes HDPE material characteristics. Please refer to table 5 above for dimensions. (D&M Plastics Incorporated, n.d.).

Table 8 - High Density polyethylene material characteristics

High Density Polyethylene	
Properties	Values
Density (g/cm^3)	0.941-0.965
Melting point (˚C)	126
Tensile Strength (N/mm^2)	0.20 – 0.40
Thermal Coefficient of Expansions (10^{-6})	100-220
Max continued use temperature (˚C)	65
Modulus of Elasticity (GPa)	0.8

After compiling the design specifications above it was agreed that for outdoor usage in urban areas these materials would produce the most resilient product with an aesthetic appeal. The materials discussed above will be the materials that we would proceed with once in production.

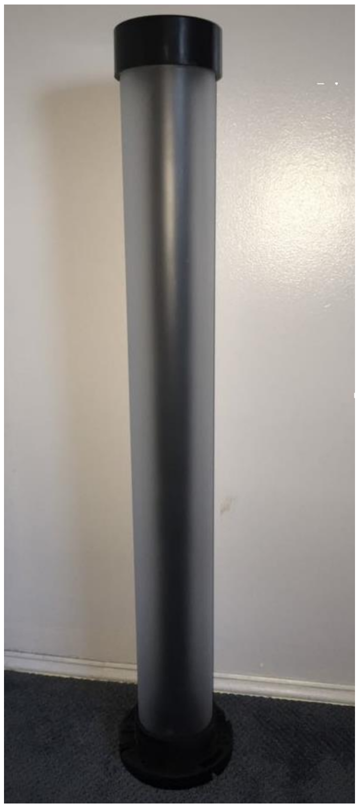


Figure 13 - Proof-of-concept mechanical prototype

7. Digital Potentiometer

As mentioned in section 3.3 that the AMR circuitry will have falsing scenarios due to earth's magnetic field and re-trimming the potentiometer to re-center to 2.5V bias value when no vehicle is present, didn't work as expected. So the alternative solution was to use a digital potentiometer to achieve the same result in the absence of vehicles. The digital potentiometer that parkolite Ltd will be using their product is one of the X9c series and it consists of wiper switches, resistor array, non-volatile memory and control section.

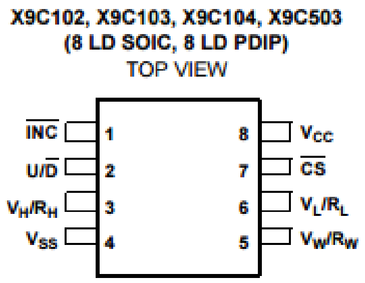


Figure 14 - Digital Potentiometer top view

The above figure shows the top view of the digital potentiometer. Our main objective is to control the position of the wiper movement, which will be controlled by pin number 1, 2 and 7. We will be monitoring the position of the wiper through pin number 5, which is the wiper terminal and is equivalent to the movable terminal of a mechanical potentiometer and the position of the wiper within the array is determined by control inputs (pin 1, 2 and 7) in this case.

7.1 Digital Potentiometer Pin descriptions

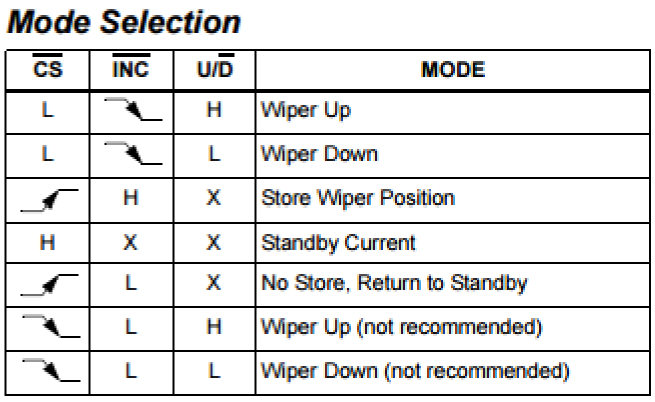
Wiper movement is controlled by three pins and their description is explained in the table below

Table 9 - Control Inputs Pin Description

Digital Potentiometer I/O	
Pin Number	Description
1	Pin 1 is the increment pin, which is negative edge triggered. Toggling this pin will result in the wiper movement and it will either increment or decrement the counter in the direction indicated by pin number 2.
2	Pin 2 is the up and down, which controls the direction of the wiper movement and if counter is incremented or decremented
7	Pin 7 is the chip select pin and this pin is used whenever we want the chip to modifiable or not. Chip is selected when this pin is set LOW.

7.2 Digital Potentiometer Programming

Digital potentiometer is programmed using arduino UNO in C/C++ environment, Figure below shows the wiper mode selection depending on the state of control inputs



Mode Selection

$\overline{CS}$	$\overline{INC}$	U/$\overline{D}$	MODE
L	↓	H	Wiper Up
L	↓	L	Wiper Down
↑	H	X	Store Wiper Position
H	X	X	Standby Current
↑	L	X	No Store, Return to Standby
↓	L	H	Wiper Up (not recommended)
↓	L	L	Wiper Down (not recommended)

Figure 15 - Wiper Mode Selection

Initially the chip select pin is set HIGH as we do not want to modify it before checking the Wiper terminal voltage, If it is already 2.5V or very close to 2.5V (0.05 less or more), digital potentiometer does not need to be programmed. If value is way off 2.5V mark, chip select pin will be set to low so that the chip can be programmed. As evident from the table above HIGH to LOW transitions on the increment pin will either increment a counter or decrement it, then next Arduino code will check conditions of pin 2 to check if it is high or low. If it is HIGH wiper will move up, if LOW wiper will move down, we do not need to store the wiper position for the objectives of our task since 2.5V reading can be achieved by programming the wiper to move up or down until that value is obtained. More detailed functionality of the control inputs are sgown in the figure below.

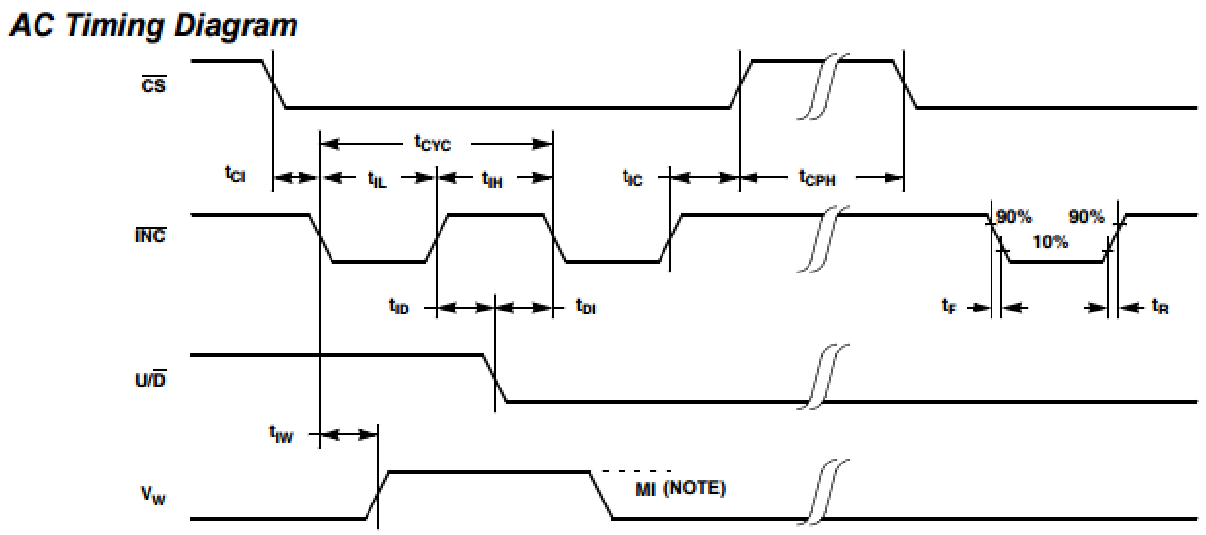


Figure 16 - AC Timing Diagram Showing Wiper Dependence on Control Inputs

8. PCB/Circuit Design

The circuit consists of three fundamental blocks: the sensor and its supporting circuitry, the embedded microcontroller and Wi-Fi module, and the power elements, which consist of three power FETs for directing current to the LEDs, and a regulator.

8.1 Sensor and supporting circuitry

This circuitry is shown in Figure 4 of the previous pages with one small difference which will be discussed shortly. It consists of a HMC1021S single-axis AMR sensor which puts out a differential voltage linearly proportional to the magnetic field present around it. The signal, which is on the order of millivolts, is amplified by an AD623 instrumentation amplifier which multiplies the signal by 500. Since the AMR sensor is always experiencing the Earth's magnetic field, there is always a non-zero voltage on the output of the amplifier. This is desirable, and, in fact, the amplifier is desired to be centered close to 2.5V (with no applied magnetic field) to allow for maximum output swing within the power rails of 0 and 5V. However, due to the varying nature of the Earth's magnetic field, temperature effects, as well as additional offset effects within the AMR sensor, the output of the amplifier will never be exactly 2.5V, and then one must call into play the trimmer potentiometer X3 on the REF pin of the AD623. Herein lies the difference between the circuit in Figure 4 and the actual circuit: we have replaced the trimmer potentiometer X3 with a digital potentiometer X9C102. This allows us to, using the microcontroller, adjust the digital potentiometer, which allows us to readjust the output of the amplifier to 2.5V at any time we wish due to the effects of the straying Earth's magnetic field and drifting temperature. Finally, the output of the amplifier is fed into a window comparator consisting of two LM393 comparators, which will detect if the amplifier output has strayed a fixed amount away from 2.5V (set by resistor R9), and generate a falling edge on its output, which is fed to the interrupt pin of the microcontroller to indicate the detection of a vehicle.

There is one more block of supporting circuitry necessary for this sensor: a degaussing charge pump circuit (See Figure 12). Powerful magnets (such as magnetized hand tools) can cause the AMR sensor to latch into a state where all of its magnetic dipoles become aligned in one direction, which causes inaccurate readings of external magnetic fields. As such, the AMR sensor IC has internal coils which will reset, or degauss, these magnetic dipoles when a spike of approximately 1A of current flows through them. Rather than draw this 1A directly from our power source, an intermediary charge pump is used. Normally Vsr is high, during which time the 5V source gradually fills capacitor C3 with charge. When Vsr goes low, the capacitor rapidly discharges to create a positive current spike through the IC coils modeled by the set/reset strap resistor in the diagram. While the positive current spike is occurring, a second capacitor is charged (capacitors C1 and C2). When Vsr goes back high, these capacitors discharge through the coils in a direction opposite to the first discharge, creating a negative spike of current. This thoroughly degausses the AMR sensor. IRF HEXFETs are required for their excellent Rds (on) and switching parameters.

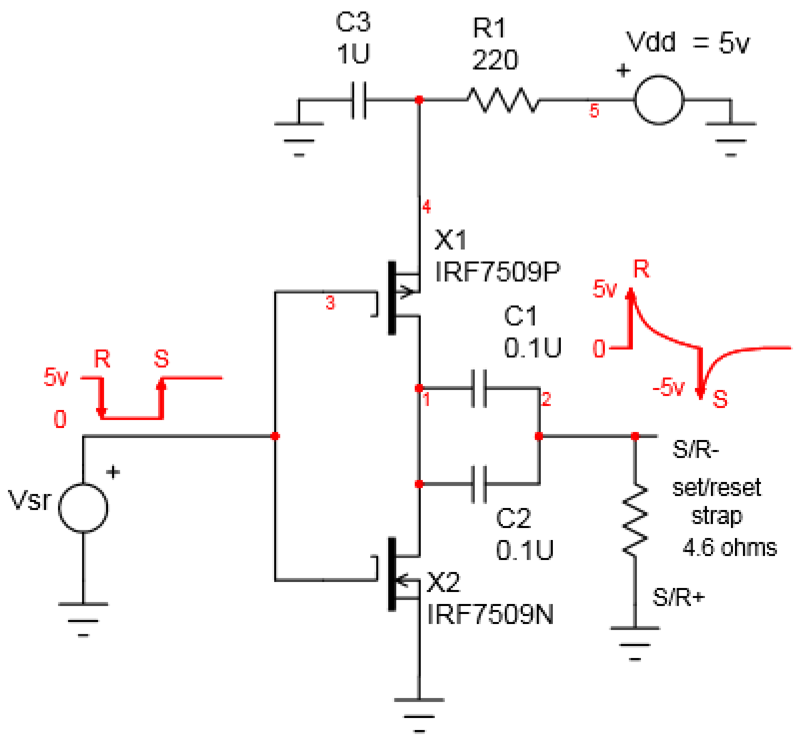


Figure 17 - Charge Pump for Degaussing the AMR Sensor

8.2 Microcontroller and Wi-Fi module

The microcontroller we are using is an ATMEGA328P that is to be programmed via the Arduino IDE and then placed into a socket on our PCB. Additionally, we are using a ESP8266 Wi-Fi module that is also to be programmed via the Arduino IDE, and then placed into our PCB via header pins.

The responsibilities of the microcontroller are as follows:

- When no car is present, periodically read the output voltage of the amplifier and ensure it is at 2.5V by adjusting the digital potentiometer
- Periodically degauss the AMR sensor
- Detect when a car has arrived by means of an external interrupt pin
- Control the color of the LEDs by means of three PWM pins for each color RGB based upon correct application of parking bylaws

The responsibilities of the Wi-Fi module are as follows:

- Read the current status of the RGB pins to determine the color status of the parking meter, and send this data to an online cloud

It should be noted that the GPIO pins on the Wi-Fi module only accept 3.3V, whereas the RGB pins of the microcontroller as 5V. As such, three intermediary level shifters are used to step 5V to 3.3V. These consist of a single NMOS FET and a couple resistors each.

8.3 Power elements

The power elements are a single 5V regulator to supply power to most of the circuitry on board (the ESP8266 Wi-Fi module has an onboard 3.3V regulator), and three power FETs to sink 12V through each respective LED color (RGB).

8.4 PCB Design

The PCB is a simple, rectangular, 2-layer board that is 3.05x2.00" in size. It was constrained in size by the fact that it needed to fit within the light fixture "pipe", and, as such, the Wi-Fi module needed to be raised into the air via header pins, which allowed other components to be placed underneath it. Figure 13 shows the PCB as designed using EAGLE CAD. The rectangular outline on the right-most side is the Wi-Fi module, and underneath it is the level-shifting circuitry as well as a reset pushbutton. To its immediate left is where the microcontroller will fit. On the top-left are all the SOIC-8 packages for the AMR sensor, instrumentation amplifier, comparator, NMOS & PMOS HEXFETs for the degaussing circuit, and digital potentiometer. Below these are four DPAK packages: the 5V regulator and three power FETs. Below these is where wires are soldered that go off-board to the LED strips as well as our power source. Finally, at the top is a long debug header that is routed to various critical signals. Mounting holes are placed at each corner.

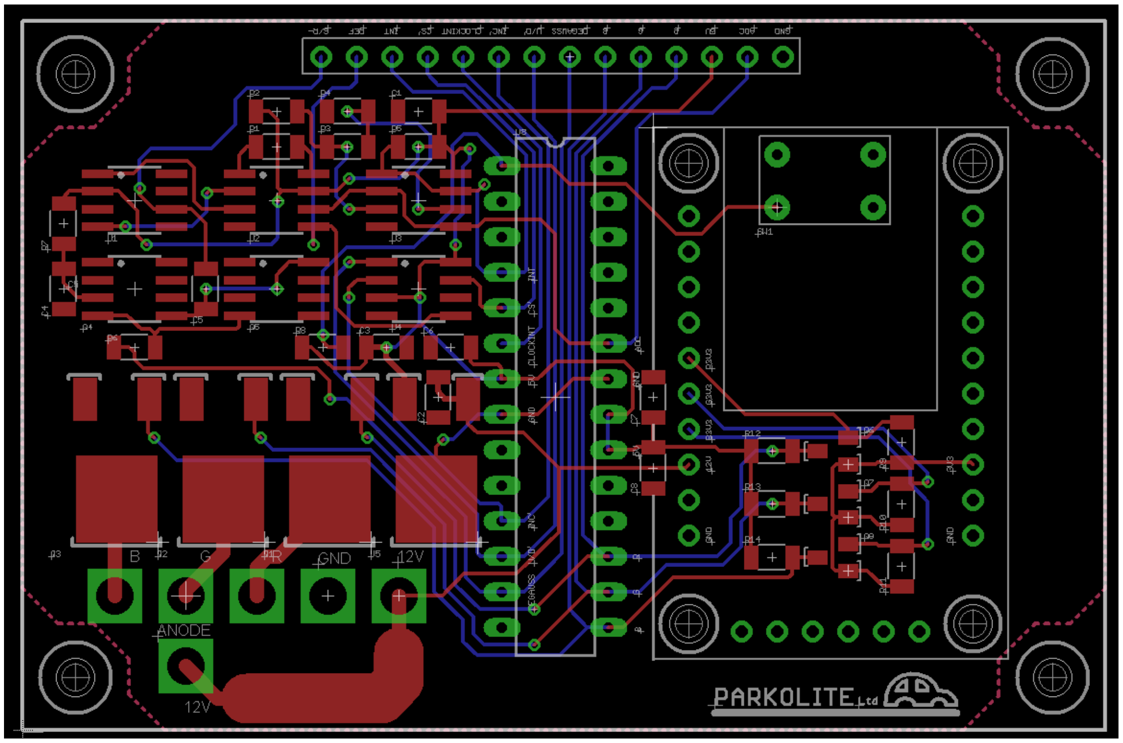


Figure 18 - PCB Design for All Ci

9. Software/App

As it can be seen from the top right corner in Figure 18 there is an empty looking area. This area has been designated for the wifi module. This wifi module has been put in place for future provisions for a simple indicator application for mobile phones. Although, the intent as of now is just to produce the hardware that actually lights up and physically tells you if you are violating or not. Eventually the consumer will be able to sign in at each pole if they have the app installed on their phone.

The app is in the preliminary stages of development and may not be able to be rolled out before the end of this year. As Parkolite continues to evolve the API into a fully functioning production ready product the app will be developed simultaneously.

For the purpose of engineering science 440 the idea will be to produce a very simple app that can communicate with the on board micro controller. When the light changes colours the app will be notified that the light has changed and the status of the current parking situation will be displayed on the screen. Although, the plan is to be able complete a simple version of this app in the following weeks it may or may not be a part of the final demonstration. As of right now, the plan is to be able to complete the API and have it functioning with up to 98% efficiency. Once the proof-of-concept has satisfied all of the preset requirements the app will be given further attention.

10. User Interface

The figure below shows a sample of what the driver would see as he or she is driving down the street. There are several instances where the driver has the opportunity to park. For example the green light on the right.

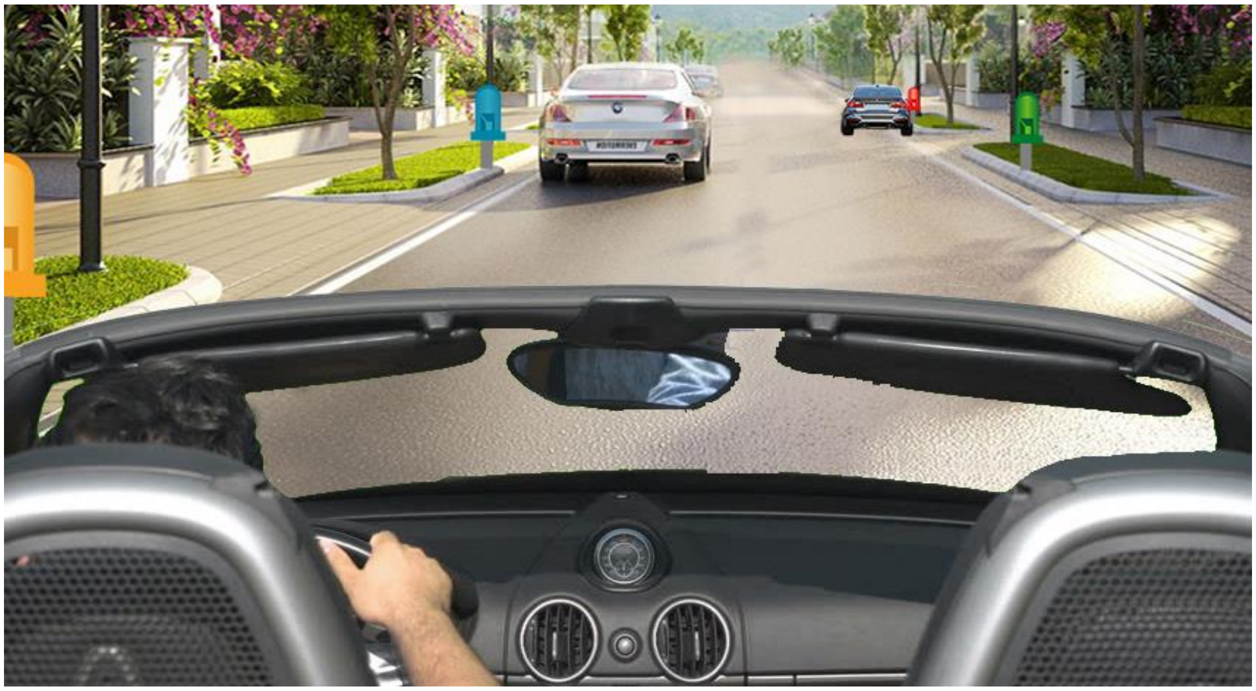


Figure 19 - Drivers Perspective

Although the user interface is not interactive it gives any user a ton of feedback through the simple use of lights. It uses simple psychotically rules, for example, red means no or not allowed. Yellow, means about to expire or about to infringe. And green can mean anything from vacant to 'go'. By using these simple everyday rules that are internationally recognized this product has the ability to be interfaced on any street anywhere in the world. With the exception to the colour blue. The reason blue was assigned the action "occupied" because red meant infringed or overstepped or incorrectly parked. From an electrical standpoint the RGB LED can be normally coded to be blue. Therefore the intent was to be able to use the LED's internal functions.

As it can be seen from the picture above, the interface is extremely simple yet very powerful. Whether you are the driver or a parking attendant being able to identify parking spots whether they are taken or over used becomes an effortless task with the use of the alternative parking indicator.

From the parking attendants perspective he/she simple has to drive down a busy street to see which car is parked incorrectly and which car is parked fine. This will vastly increase the possibility for the city to generate profits based on people who overuse designated parking spots.

11. Test Plans

The quality and usability of each component is extremely important, because this product will be marketed to the City Vancouver. In order to ensure to reach the ultimate quality and performance in API system, there will be 3 testing phases; Unit Acceptance Testing phase, the Integration Testing phase and regression Testing Phase.

11.1 Unit Acceptance Testing – UAT Phase

First phase of testing – Unit Acceptance Testing- in which all the – mostly electrical- components of the system is being tested for their functionality.

11.1.1 Microcontroller Unit Test Plan

Several short projects will be run on the microcontroller to be able to explore its capabilities in UAT phase. ATmega328p chip will be integrated with AMR sensor, Digital Potentiometer and LED's to ensure a coherence performance between elements.

11.1.2 Digital Potentiometer Unit Test Plan

Arduino controls the digital potentiometer; it is programmed to keep the voltage around 2.5 – the preferred voltage for our project. Given the input to digital potentiometer will be larger or smaller voltages the output should be around 2.5- with given tolerance. The position of the wiper will be read by the programmed to check the accuracy and functionality of both potentiometer and the written code.

11.1.3 AMR Sensor Unit Test Plan

- Using a Red LED: LED will turn on as a ferromagnetic object gets close to it.

11.1.4 LED Unit Test Plan

- Each color will be tested by sourcing proper power.
- The intensity of LED lights will be determined by the amount of current sourced.
- Using a mechanical switch: each external trigger will change LED colors.

11.2 Integration Testing Phase

In the second phase of testing, the communication between components is being tested. We have to make sure correct values are being passed to each system, and if signals are being properly triggered by external and internal interrupts.

11.2.1AMR Sensor Integration Test Plan

AMR Sensor will send a signal to Microcontroller once it detects a ferromagnetic object. The sensitivity of the AMR Sensor will be tested, meaning not all ferromagnetic objects are favourable for our project. AMR Sensor will detect motor vehicle and send signal to Microcontroller to process.

11.2.2 Potentiometer Integration Test Plan

Potentiometer will ensure the voltage sent to microcontroller is within the preferred range – 2.5 V. If the voltage is not, then it will adjust the voltage and send the proper signal (readjusted voltage) to board.

11.2.3 Microcontroller integration Test Plan

Microcontroller is controlling all the components continuously and simultaneously; once receives signal from potentiometer, it will then actives an external trigger which will then be processed within the firmware and the output will be visibly shown in LED colors.

11.2.4 LEDs integration Test Plan

Each LED color will be laminated based on a pre-programmed case, which will be triggered by external – AMR Sensor output- and internal interrupt – time.

11.3 Regression Testing Phase

In this phase, all the integration-testing phase will be tested multiple times. Several cases of failure will also be tested.

11.4 App integration Phase Testing

This part of the test plan is solely dependent on time. If time permits Parkolite will produce a mobile application that will be able to communicate with the on board micro controller. The test plan will look something like this:

- If the light is green the app will show a bare green screen
- Once parked the light will go from green to blue. On the screen it will indicate that you have occupied the parking spot. It will also indicate that you are in a good parking status
- When you are within 15 minutes of expiration, the application will change from a blue screen to a yellow screen notifying a warning. This warning will say that the parking is about to expire
- Once the allotted time has been surpassed the screen will then turn red. Giving the user a sign that says "you are at risk of being ticketed please move your car".
- If the user leaves before the end of the parking period the phone signs off from that parking meter and black screen shows on the app saying "waiting to intercept Parkolite"

12. Concluding Remarks

Parkolite Ltd is absolutely committed to improve the parking issues across Canada by providing better interpretation of bylaws to drivers and at the same time assisting parking officers to ticket the violators. Our team of engineers is very excited to introduce their product *alternative parking indicator (API)*, which will be a pole model design – implemented on top of the curb or sidewalks.

Among all the complex bylaws and parking sign that gives drivers a hard time to interpret the signs, API is hoping to kick start the next step which evolutes these complex signs to simple form of different colours, which can be easily be interpreted and saves drivers all the confusion related to bylaws.

Design specification document builds up on the functional specification document and all the details and procedure mentioned in this document has been or will be practiced by ParkoLite members to ensure high quality of the product. Various electrical and mechanical components that are mentioned throughout this document have been analysed to ensure proof-of-concept model. Lastly, test plans mentioned in this document will be carried out throughout the integration part and in the final stages of the product to check the functionality of the product.

References

Atmel Corporation. "*Atmel datasheet.*" 2011: web. 12 Oct. 2015 <http://www.atmel.com/Images/doc8025.pdf>

Vancouver 24 Hrs. "*Surrey Parking in Chaos.*" 2014: web. 18 Oct. 2015 <http://vancouver.24hrs.ca/2014/07/16/surrey-parking-in-chaos>

LEDLIGHTSWORLD. "*Flexible Led Strip Light Specification*." 2015: web. 12 Oct. 2015 <http://www.ledlightsworld.com/datasheet/Specification-of-Flexible-LED-Strip-www.ledlightsworld.com.pdf>

ANALOG DEVICES. "*Integrated AMR Angle Sensor and Signal Conditioner.*" 2014: web. 12 Oct. 2015 <http://www.analog.com/media/en/technical-documentation/data-sheets/ADA4571.pdf>

Honeywell. "*HMC1001/1002 SPECIFICATIONS*." August 2008: web. 12 Oct. 2015 <http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Missiles-Munitions/HMC_1001-1002-1021-1022_Data_Sheet.pdf>

MacKay housing. "*MKH 4500 specification*." 2004: web. 12 Oct. 2015 <http://www.mackaymeters.com/Portals/0/Documents/Brochures/MKH4500_L.pdf>

Engineering Policy Guide. "*Barrier Curb*. "April 2011: web. 12 Oct. 2015 http://epg.modot.org/files/0/01/620.2.23.jpg

AZO materials . (2015). *Tungsten Carbide - An Overview*. Retrieved from azome: http://www.azom.com/properties.aspx?ArticleID=1203

City of Toronto . (2015). *Parking Activity 2014.* Toronto: City of Toronto.

City of Vancouver. (2012, July 31). *Streets and Transportation*. Retrieved from Vancouver: http://vancouver.ca/streets-transportation/claim-a-towed-vehicle.aspx

City of Vancouver. (2015). *Vancouver 2015 Budget.* Vancouver: City Of Vancouver.

Collier Internation . (2012, 10 02). *Parking Rates Rise with Calgary Leading Canadian Cities*. Retrieved from CBC: http://www.cbc.ca/news/canada/parking-rates-rise-with-calgary-leading-canadian-cities-1.1156681

D&M Plastics Incorporated. (n.d.). *Polyethylene all about plastic moulding*. Retrieved from Plastic Moulding: http://www.plasticmoulding.ca/polymers/polyethylene.htm

Government of Canada. (2015, June 29). *Motor Vehicle Registrations*. Retrieved from Statistics Canada: http://www.statcan.gc.ca/tables-tableaux/sum-som/l01/cst01/trade14a-eng.htm

Mulgrew, I. (2014, Septemeber 20). Vancouver's bylaw ticket system raises questions about fairness. *Vancouver Sun*. Retrieved from http://www.vancouversun.com/opinion/columnists/Mulgrew+Vancouver+bylaw+ticket+system+raises/10219655/story.html

Web Civil. (2015). *Steel Properties*. Retrieved from web Civil: http://www.webcivil.com/frmsteelproperty.aspx

Note: ParkoLite members created all figures and tables and they are property of ParkoLite Ltd.

Appendix – Controller Programming Unfinished

```
//THE FOLLOWING ARE FOR THE LEDS AND BYLAWS
int r = 9;
int g = 10;
int b = 11;
// for interrupt use input pin arduino Pin 2
int inputpin = 2; // this is also interrupt 0;

int black [3] = {0,0,0};
int white [3] = {100,100,100};
int red[3]    = { 100, 0, 0 };
int green[3]  = { 0, 100, 0 };
int blue[3]   = { 0, 0, 100 };
int yellow[3] = { 100, 60, 0 };

// Set initial color
int redVal = red[0];
int grnVal = green[1];
int bluVal = blue[2];

int wait = 10;      // 10ms internal crossFade delay; increase for slower fades
int hold = 0;       // Optional hold when a color is complete, before the next crossFade
int DEBUG = 1;      // DEBUG counter; if set to 1, will write values back via serial
int loopCount = 60; // How often should DEBUG report?
int repeat = 1;     // How many times should we loop before stopping? (0 for no stop)
int j = 0;          // Loop counter for repeat

// Initialize color variables
int prevR = redVal;
int prevG = grnVal;
int prevB = bluVal;

int brightness = 0;
int fade = 5;
int value = 0;

volatile byte LED_state = LOW;

// THE FOLLOWNG ARE FOR THE CLOCK
int clockInt = 1; // digital pin 3 = interrupt 1 ;
int masterClock = 0; // counts rising edge clock signals
int seconds = 0; // seconds variable
int minutes = 0; // minutes tracker variable
int hours = 0; // hours tracker
int state = LOW;
int oldstate = HIGH;

// THE FOLLOWING ARE FOR THE DIGIPOT
int CS = 5; // used for digitpot code - these values have to be different
int INC = 5; // used for digipot code - this value as well
int UD = 7; // this value has to be different too
const float VoltageTolerance = 0.05; //V


void setup()
{
  // put your setup code here, to run once:
  // this code is to initaliaze the LEDs and input pin
  //THE FOLLOWING INITALIZATIONS ARE FOR THE LEDS AND BYLAWS
        pinMode(inputpin, INPUT);

        pinMode (r, OUTPUT);
        //digitalWrite (r, OUTPUT);

        pinMode(g, OUTPUT);
        digitalWrite(g, OUTPUT);

        pinMode (b, OUTPUT);
        //digitalWrite(b, OUTPUT);

        // THE FOLLOWING INITIALIZAIONS ARE FOR THE CLOCK

        attachInterrupt (clockInt, clockCounter, CHANGE);
        analogReference (DEFAULT);
        Serial.begin(9600);
        analogWrite (6,127); // this starts our PWM 'clock' with 50% duty cycle off of pin 6
       PWM

        // THE FOLLOWING INITALIZATIONS ARE FOR THE DIGITPOT
         pinMode(CS, OUTPUT);      // CS
         pinMode(UD, OUTPUT);      // U/D
         pinMode(INC, OUTPUT);     // INC
         pinMode(A0, INPUT);       // ADC

         digitalWrite(CS, HIGH);    // Set the digipot to be unmodifiable
         digitalWrite(INC, HIGH);  // Set INC to high, later changing it to low will increment
       the wiper of the digipot
}

void loop() {
  // put your main code here, to run repeatedly:
  // the following code is being looped for the clock
  //*******************CLOCK*************************
  timeKeeper();
  //*******************LED/BYLAW*********************

  ISRPB(inputpin);


}
//********************CLOCK FUNCTIONS START**********
void clockCounter()
{
  masterClock++; //each clock rise will add '1' to masterClock
  if (masterClock == 1960) // 490 Hz reached
  {
   seconds++;  // after one 490Hz cycle add '1' second
   masterClock = 0; // reset master clock after 1 second is reached
   state = !state;
  }
  return;
}

void printTime()
{   // this function will print the following format on the Serial COM
    // Time = 00:00:00 // and will increment accordingly
          Serial.print("  Time = ");

          zeroit(hours);
          Serial.print(hours);

          Serial.print(":");
          zeroit(minutes);
          Serial.print(minutes);

          Serial.print(":");
          zeroit(seconds);
          Serial.println(seconds);

   return;
}

void zeroit(int value)
{
  if (value < 10)
```

```
 Serial.print("0");
 return;
}

void timeKeeper()
{
 if(seconds == 60)     // NOW GETTING IN TO REAL TIME KEEPING
   {
    if (minutes == 60)// set to 60
    {
     hours++;
     minutes = 0;
     seconds = 0;
    }
   else
    {
     minutes ++;         // increment minutes by 1
     seconds = 0;        // reset the seconds variable
    }
  }

   if (state != oldstate)
   {
    oldstate = state;
    printTime();
   }
}
//******************CLOCK FUNCTIONS END***************************

// ****************DIGIPOT FUNCTIONS******************************

void AdjustPotentiometer(int amount, boolean direct)
{
   digitalWrite(CS, LOW);       // Activate the chip to make it modifiable
   digitalWrite(UD, direct);    // Select the direction to move the wiper
   for(int i=0; i<amount; i++)
   {
    digitalWrite(INC, LOW);    // Increment the wiper with a high->low transition on INC
    digitalWrite(INC, HIGH);   // Bring INC back to high
   }
 digitalWrite(CS, HIGH);      // Deactivate the chip from being modifiable
}

float ReadVoltage(){
 int sensorValue = analogRead(A0);
```

```
  float voltage = sensorValue * (5.0 / 1023.0);
  //Serial.println(voltage);
  return voltage;
}

void SetDigipot(float targetVoltage){
  boolean breakflag = 0;     // This variable is used to break the following while loop
  float voltage;

  while(breakflag == 0){
    voltage = ReadVoltage();
    if(voltage < (targetVoltage+VoltageTolerance) && voltage > (targetVoltage-
VoltageTolerance)){
      breakflag = 1;       // Break the loop
    }
    else if(voltage > targetVoltage){
      AdjustPotentiometer(1, 0); // Increment the wiper by amount=1 in direction=0 (down)
    }
    else if(voltage < targetVoltage){
      AdjustPotentiometer(1, 1); // Increment the wiper by amount=1 in direction=1 (up)
    }
  }
}
//*****************DIGIPOT FUNCTIONS END********************************

//*****************LED FUNCTIONS BEGIN**********************************

int calculateStep(int prevValue, int endValue)
{
  int step = endValue - prevValue; // What's the overall gap?
  if (step) {                   // If its non-zero,
    step = 1020/step;            //   divide by 1020
  }
  return step;
}

int calculateVal(int step, int val, int i)
{

  if ((step) && i % step == 0) { // If step is non-zero and its time to change a value,
    if (step > 0) {            //   increment the value if step is positive...
      val += 1;
    }
    else if (step < 0) {        //   ...or decrement it if step is negative
      val -= 1;
```

```
   }
  }
  // Defensive driving: make sure val stays in the range 0-255
  if (val > 255) {
    val = 255;
  }
  else if (val < 0) {
    val = 0;
  }
  return val;
}

void crossFadeG(int color [3])
{
  int G = (color[1] * 255) / 100;
  int stepG = calculateStep(prevG, G);

  for (int i = 0; i <= 1020; i++)
  {
    grnVal = calculateVal(stepG, grnVal, i);
    analogWrite(g, grnVal);
    delay(wait); // Pause for 'wait' milliseconds before resuming the loop

  }

  prevG = grnVal;
  delay(hold); // Pause for optional 'wait' milliseconds before resuming the loop
}

void crossFadeB(int color [3])
{

  int B = (color[2] * 255) / 100;
  int stepB = calculateStep(prevB, B);

  for (int i = 0; i <= 1020; i++)
    {
      bluVal = calculateVal(stepB, bluVal, i);
      analogWrite(b, bluVal);
      delay(wait); // Pause for 'wait' milliseconds before resuming the loop

    }

  prevB = bluVal;
  delay(hold); // Pause for optional 'wait' milliseconds before resuming the loop
```

```
}

void crossFadeR(int color [3])
{
  int R = (color[0] * 255) / 100;
  int stepR = calculateStep(prevR, R);

  for (int i = 0; i <= 1020; i++)
  {
    redVal = calculateVal(stepR, redVal, i);
    analogWrite(r, redVal);   // Write current values to LED pins
    delay(wait); // Pause for 'wait' milliseconds before resuming the loop

  }
  prevR = redVal;
  delay(hold); // Pause for optional 'wait' milliseconds before resuming the loop
}

void crossFadeY(int color [3])
{
   // Convert to 0-255
  int R = (color[0] * 255) / 100;
  int G = (color[1] * 255) / 100;
  int B = (color[2] * 255) / 100;

  int stepR = calculateStep(prevR, R);
  int stepG = calculateStep(prevG, G);
  int stepB = calculateStep(prevB, B);

  for (int i = 0; i <= 1020; i++)
  {
    redVal = calculateVal(stepR, redVal, i);
    grnVal = calculateVal(stepG, grnVal, i);
    bluVal = calculateVal(stepB, bluVal, i);

    analogWrite(r, redVal);   // Write current values to LED pins
    analogWrite(g, grnVal);
    analogWrite(b, bluVal);

    delay(wait); // Pause for 'wait' milliseconds before resuming the loop
  }
  prevR = redVal;
  prevG = grnVal;
  prevB = bluVal;
  delay(hold); // Pause for optional 'wait' milliseconds before resuming the loop
```

```
}

void blink(){LED_state =! LED_state;}

void ISRPB (int interrupt){
  switch (LED_state){
    case 1:
    if (interrupt = LED_state)
    {
        crossFadeB(blue);
        digitalWrite(g, LOW);
    }
    case 2:
    if (seconds == 20)
    {
      crossFadeY(yellow);
      digitalWrite(b,LOW);
    }
    case 3:
    if (seconds == 40)
    {
      crossFadeR(red);
      digitalWrite(g,LOW);
      digitalWrite(b,LOW);
    }

  }

}
```