IoT Green Corridor

R Ramapriya
ramapriya288@gmail.com

***Abstract*—This is a system that combines the existing trivial traffic signal lights with sensors, which are capable of synchronizing with each other, and take certain decisions on the switching of lights as per the given set of conditions. Synchronizing of sensors is possible only by creating a network among them. Hence, the sensors are mutually connected by means of internet using an IoT (Internet of Things) network. The Internet of Things (IoT) is a new and evolving concept that provides connectivity to the Internet via sensing devices to achieve intelligent identification and management in a heterogenous connectivity environment. The aim is to design a prototype of traffic junctions which can sense the presence of an Emergency Vehicle (EMV) within a threshold distance and change the traffic flow, so that the EMV passes the signal within less time and uninterrupted. This very prototype consists of sound sensors (amplified by IC LM 358), which are connected to a circuit comprising of PIC 12F683 microprocessor, that senses the presence of the EMV by detecting the frequency of its siren. Based on the output of this frequency detecting circuit, the other sound sensors placed at the respective lanes start sending data to an Intel Galileo Board, which is a microcontroller, that takes decisions on switching of traffic lights in the junction, by comparing the inputs given by the sound sensors. As soon as the EMV leaves the junction, The Galileo Board sends a signal to a Linkit ONE board (an IoT development board) which sends a value to the Cloud, which in this case is the Ubidots cloud computing platform, signalling that the EMV has passed the junction. As soon as the cloud receives this signal an event-based action occurs, which sends a signal to the next junction, according to the pre-defined path of the EMV, indicating that it is approaching the junction. This next junction consists of a similar setup of microcontrollers and sensors, which repeat the process. This occurs at each traffic junction till the EMV reaches its destination.**

***Index Terms*—Emergency Vehicles, IoT, Green Corridor, Goertzel's Algorithm**

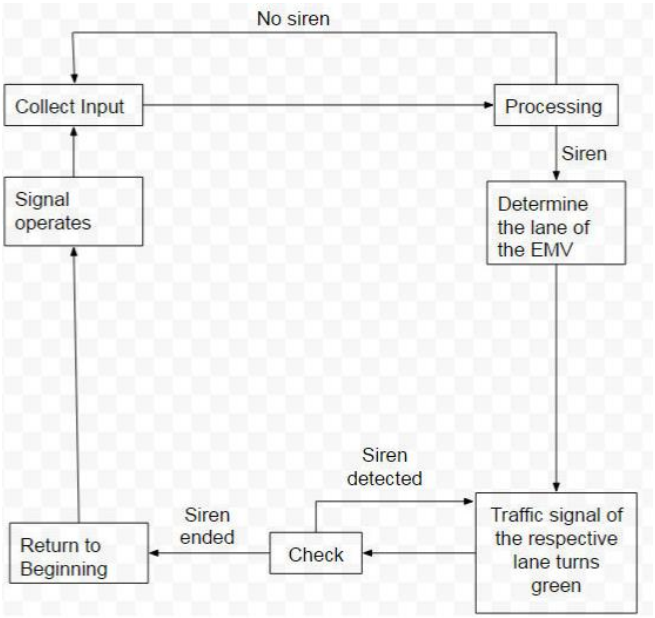
I. INTRODUCTION

Every other day we are stuck in a jam, or we wait at a red signal and hear the siren of an Emergency Vehicle (EMV), such as an ambulance or a fire engine, going off behind us. But if there is no traffic police to take charge of the situation, we are helpless, and the EMV is forced to wait till the light goes green. Every year in India alone, several thousands of people lose their life because they were not given proper treatment on time, one of the prime reasons being the EMVs not able to reach their destination fast enough. The one thing that crosses our mind is, if only there was a system in place that would take care of this automatically, so people have no choice but to give the ambulance a free and clear path. This is exactly what this project aims to do. There is a mechanism in place at every traffic junction that detects the presence of an EMV and provides it a clear path. This mechanism can be efficient only if human-human or human-machine interaction is minimized, and hence the term IoT comes into play. IoT in layman's terms is the mutual interaction of objects with each other, causing event-based actions to take place. These objects may be sensors, microcontrollers, etc., and they communicate wirelessly thus creating a network, thus forming the Internet of Things The Internet of things (IoT) is the network of physical devices, vehicles, buildings and other items - embedded with electronics, software, sensors, actuators and network connectivity that enable these objects to collect and exchange data with the manufacturer, operator and/or other connected devices. Internet enabled objects, together with web services that interact with these objects, form this network. The Internet of Things enables objects to be sensed and controlled remotely across the existing network infrastructure, creating opportunities for more direct integration between the physical world and computer-based systems, and resulting in enhanced efficiency, accuracy and economic benefit. These devices collect useful data with the help of various existing technologies and then autonomously flow the data between other devices. In this project, a network of sensors is created at each junction. We all know that an ambulance or fire engine or any EMV for that matter has a characteristic siren. A sound sensor is connected in a circuit with a microprocessor programmed to detect this specific frequency. As soon as this frequency is detected, more sound sensors placed along the sides of the road are used to detect which lane the ambulance is in. Once this is known, the respective signal light goes green, and the others red, till the ambulance has crossed the junction. The sensors communicate wirelessly through the cloud. Each sensor sends its registered readings to the cloud and this data is analyzed and processed in the cloud. It then sends back specific data which causes the action of changing the lights to take place. The traffic from other lanes must be stopped until the emergency vehicle leaves the junction, hence, creating a green corridor for the EMV.

II. DESIGN

The frequency sensor consists of a simple microphone with an amplifier connected in a simple circuit that processes the input collected by it. Following are the events that occur one after the other that lead to the changing of the signal light. The microphone of the frequency sensor keeps collecting input. The signals follow their default pattern. This input is processed by the PIC12F683 microprocessor using discrete Fourier transform. If the siren is not detected it repeats the process till a siren is detected. If a siren is detected, the readings from the amplitude sensors are then analyzed and the greatest one is found. Based on this the lane of the EMV is detected. Accordingly, the corresponding traffic light turns green and the others turn red. The light that turns green is the

one directly opposite the EMV. The frequency sensor continuously keeps checking if the conditions remain the same to keep the signal from changing till the ambulance has passed. If the siren is still present, it keeps the signal in the same state. If the siren is no longer loud enough to sense, indicating that the EMV has passed the junction, The default pattern is resumed. Also, as soon as it leaves the junction a value 1 is sent to a variable in Ubidots, the cloud computing platform used in this project. An event created in Ubidots then sets another variable to This is sent to the next junction to inform it that the EMV is approaching.



This is the basic System Block Diagram

III. HARDWARE

Intel Galileo Board: It is a first in a line of Arduino-certified development boards based on Intel x86 architecture and is designed for the maker and education communities. Intel Galileo combines Intel technology with support for Arduino ready-made hardware expansion card (called "shields") and the Arduino software development environment and libraries. The development board runs an open source Linux operating system with the Arduino software libraries, enabling re-use of existing software called "sketches". Intel Galileo can be programmed through OS X, Microsoft Windows and Linux host operating software. The board is also designed to be hardware and software compatible with the Arduino shield ecosystem.

Linkit ONE development board: The Linkit ONE development platform is an open source, high performance board for prototyping wearables and IoT devices. It is based on the world's leading SoC for wearables, MediaTek Aster (MT2502) combined with high performance Wi-Fi (MT5931) and GPS (MT3332) chipsets to provide you with access to all the features of MediaTek Linkit. It also provides similar pin-out features to Arduino boards, making it easy to connect various sensors, peripherals and Arduino shields. Linkit ONE is an all-in-one prototyping board for IoT/ wearables devices. Integrating GSM, GPRS, Wi-Fi, GPS, Bluetooth features into a basic Arduino form factor.

Grove Sound Sensor: A Grove - Sound sensor can detect the sound strength of the environment. The main component of the module is a simple microphone, which is based on the LM358 amplifier and an electret microphone. This module's output is analog and can be easily sampled and tested by any Arduino compatible microcontroller board

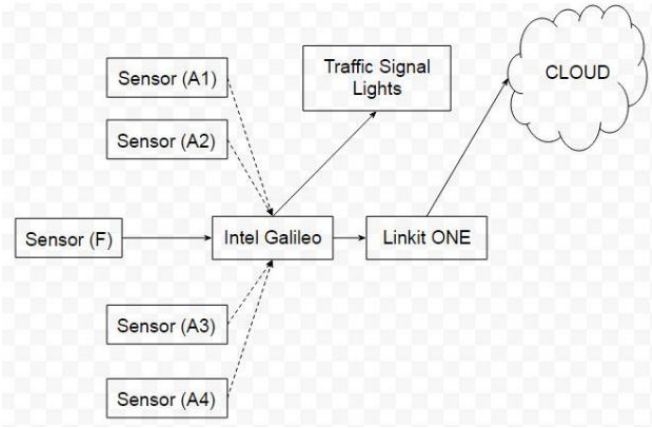
PIC12F683 microprocessor: These are the Low Pin-Count (8) PIC Flash Microcontroller products that offer all of the advantages of the well-recognized midrange x14 architecture with standardized features including a wide operating voltage of 2.0-5.5 volts, on-board EEPROM Data Memory and nanowatt Technology. Standard analog peripherals include up to 4 channels of 10-bit A/D, an analog comparator module with a single comparator, programmable on-chip voltage reference and a Standard Capture/Compare/PWM (CCP) module

Base Shield: The Grove base shield plugs into all Arduino compatible boards and is the foundation of the Grove system. All I/O ports of the controller board are exposed and adapted into Grove connectors which include digital I/O, analog I/O and specialized ports (I2C, SPI, UART).

Final Stage

IV Hardware Block Diagram

A frequency sensor (consisting of a simple microphone in a circuit) first detects the presence of a sound of frequency 750 Hz. If it does detect a sound of this frequency, the analog inputs from the four amplitude sensors, each placed on the left side of the road is read into the Galileo. The Galileo then checks to see which sensor records the highest value, and accordingly gives a high output to the respective pin. This high output is sent to one of three analog input pins of linkit one. Depending on which pin turns high, the respective traffic signal turns GREEN and the others turn RED. This occurs as long as the readings of that amplitude sensor remains the highest. When the readings of all the sensors are then below a threshold, indicating that the ambulance has crossed the junction, the traffic signals continue to change as per the default cycle



The frequency detector circuit can be used to detect the presence of a certain frequency within an analog signal, such

as an audio signal. This circuit is based on 8 pin PIC 12F683 microprocessor. Only a few additional resistors and capacitors are needed to complete a circuit that will accept an analog signal and drive a microprocessor output pin high when the selected frequency is present in the signal. The steps that follow detail operation of the circuit and the program that runs on the processor. A description of the digital signal processing algorithm used by the PIC is included. The code for programming the processor is included as a .hex file and also the source code .as file is also provided. The operation of the circuit and function of each component is described This circuit can easily be built on a breadboard. You will need a programmer for PIC microprocessors. In this case we use PICkit 3 to program the microprocessor. We program the PIC processor using a PICkit 3 programmer and for this we need MPLABX IDE and MPLAB IPE software's.

IV. GOERTZEL'S ALGORITHM

The Goertzel algorithm is a signal processing algorithm which is used for detecting a single frequency. It is derived from the Fourier transform. The algorithm acts as a very narrow band pass filter. It produces a very sharp response to frequencies within the pass band, and a much lower response for frequencies outside the pass band. The minute details of the Goertzel algorithm won't be covered here. The algorithm loop samples the input using the microprocessor built in A/D converter. The necessary mathematical operations are performed in the time between successive samples

$$\text{Cosine Coefficient} = \cos\left(2\pi\frac{f_{target}}{f_{sample}}\right)$$

$$\text{Sine Coefficient} = \sin\left(2\pi\frac{f_{target}}{f_{sample}}\right)$$

$$W_n = 0.54 + 0.46 \cdot \cos\left(2\pi\frac{n}{N}\right)$$

$$X_n = X_{n\_sample} \cdot W_n$$

$$Y_0 = X_n + Y_1 \cdot \text{Cosine Coefficient} - Y_2$$

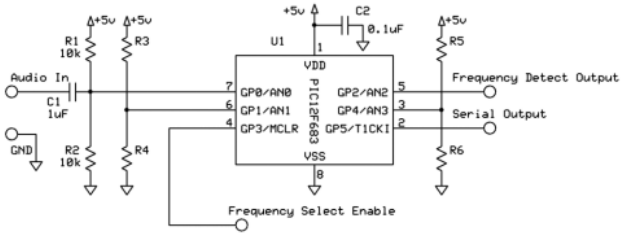
$$Real = Y_1 - Y_2 \cdot \text{Cosine Coefficient}$$

$$Imaginary = Y_2 \cdot \text{Sine Coefficient}$$

$$Magnitude = \sqrt{(Real)^2 + (Imaginary)^2}$$

Xn_sample=The latest sample from the A/D converter
Xn=The latest sample from the A/D converter multiplied by the window function.
Y0=The output value presently being computed.
Y1=The previously computed output value. (The output value from the previous iteration of the loop).
Y2=The output value from the iteration the iteration before the l previous iteration of the loop.
The figure below shows a list of the variables used in the algorithm. Refer to the flow chart in the blog to see how the Goertzel algorithm portion of the program executes on the processor. Once the algorithm has processed all the samples (the attached *.hex and *.as files use 200 samples), the real and imaginary components of the results are calculated. The real and imaginary portions are then used to compute the magnitude. The magnitude is a measure of the target frequency is present in the sampled data. The magnitude is then compared against the threshold. If the value is greater than the threshold value, the frequency is considered to be detected, and the output is set HIGH. If the magnitude is lower than the threshold, the output is set LOW. The sharpness of the filter response versus frequency is proportional to the number of samples taken. The response of the algorithm must be sharp enough that it responds to the target frequency but produces much lower response for frequencies outside the target. A value of 200 samples was found to produce a reasonably narrow response in experimentation.

V. Building the Circuit



Pin 1: Vdd (+5 volts)
Pin 2: Serial Output
Pin 3: Detection Threshold Analog Input
Pin 4: Frequency Select Enable
Pin 5: Frequency Detect Output
Pin 6: Frequency Select Analog Input
Pin 7: Biased Signal Analog Input
Pin 8: VSS (Ground)

The circuit is built on the breadboard as the following circuit diagram. The programmed PIC microprocessor is used in the circuit as shown in the schematic. The A/D converter measures voltages between 0V and the supply voltage VDD, which is 5V in this case. The analog input may be a signal that swings above and below 0 volts, so its waveform will have positive and negative portions. In order to sample the input waveform with the A/D converter, the input signal needs to be shifted. The voltage divider created by resistors R1 and R2 sets the bias at half of the 0 to VDD A/D input range. The capacitor C1 couples the AC input signal to the A/D input, so now the input waveform swings about ½ VDD instead of swinging about 0 volts. R3 and R4 form a voltage divider used to set the target frequency. The processor reads the voltage at the frequency select pin and uses the value to determine the correct coefficients needed by the algorithm to determine if the target frequency is present. R5 and R6 form a voltage divider used to set detection threshold. This allows the user to select the magnitude of the algorithm output that is necessary to make the program turn the output HIGH. The processor reads the voltage at the detection threshold pin and uses the value to determine the detection threshold. C2 is a decoupling capacitor used with the microprocessor to keep the VDD supply free of spikes

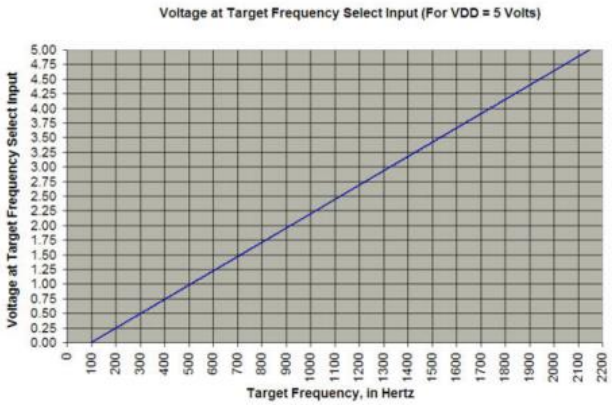
VI. Target Frequency Selection

The target frequency is selected by means of a voltage applied to the frequency select input. The voltage on this input is ready by the microprocessor's A/D converter, and from this reading the program configures itself to decode the corresponding frequency. The frequency range that can be decoded is between 100Hz and 2148Hz. The program always reads the target frequency select input after power up. After that, the target frequency select input is ignored, unless the target frequency select input enable is brought low. So, to make changes to the target frequency during program operation, the enable input must be low. It needs to be kept low for at least 100ms, as the input is checked at the beginning of the program loop. Once the input is brought high again, the program will ignore the target frequency select input, and the target frequency in effect will be that established while the enable pin was low. A voltage divider is used to set the correct input voltage on the pin to select the desired target frequency.

V_target_frequency_select_input= VDD*(Target_Frequency -100)/ 2048

VDD is the power supply voltage (usually 5 Volts). The valid range of frequencies for the program as written is 100Hz to 2148 Hz. So, for a target frequency of 750Hz, and VDD of 5 Volts, the voltage applied to the frequency select input should be:

V_target_frequency_select_input = 5*(750 -100)/ 2048 = 1.59 volts

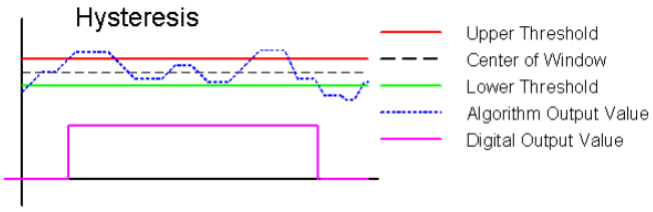


The graph shows the target frequency selected versus voltage on the target frequency select input for a VDD of 5 volts. The easiest way to set the voltage is using a voltage divider, as shown in the schematic.

The voltage at the output of a voltage divider is: Vout=VDD*[R4/(R3+R4)] The exact values of the resistors is not critical, only the ratio of their values. However, to prevent noise from upsetting the frequency select input, it is best to choose lower resistor values. For the prototype, we chose the values to be 1k and 2.2k.

VII. Detection Threshold Settings:

The detection threshold is set by means of a voltage applied to the detection threshold input. The voltage on this input is ready by the microprocessor's A/D converter, and from this reading the program sets the detection threshold



The value read from the threshold setting input establishes the center value of a hysteresis window, with the upper and lower thresholds sitting symmetrically above and below the center. The output value from the algorithm must exceed the upper threshold to cause the detect output to go high, and the output value must drop below the lower threshold before the detect output will go low. The smaller variations in output value above and below the center of the window do not cause the detect output to change. The software does not allow the detection threshold to be set all the way to zero, as this would cause the detection output to go high for any small amount of noise on the signal input. The processor uses the VDD power supply on pin 1 as the reference for the A/D converter. If a different VDD is used the output of the algorithm for a given signal input level will change proportionately. A voltage divider is used to set the correct input voltage on the pin to select the desired detection threshold

To calculate the voltage to apply to the detection threshold input to establish the minimum upper threshold required to detect an input signal of a certain amplitude at the target frequency, use the following equation.

V_dection_threshold_input=VDD*[342*V_signal_at_target_freq – 27] / 1024

So, to set the detection threshold so that the detection output goes high for an input signal of 1 volt (peak) or greater at the target frequency, for a VDD of 5 volts, the voltage at the detection threshold input should be:

V_dection_threshold_input = 5 * [342 * 1 -27] / 1024 = 1.54 Volts

The graph below shows the upper and lower thresholds of the hysteresis window versus voltage on the detection threshold input for a VDD of 5 volts. The easiest way to set the voltage is using a voltage divider, as shown in the schematic. The voltage at the output of the voltage divider is:

Vout=VDD*[R6/(R5+R6)]

The exact values of the resistors is not critical, only the ratio of their values. However, to prevent noise from upsetting the value read by the input, it is best to choose lower resistor values. For the prototype we chose the values to be 1k and 2.2k.

VIII. CONCLUSION.

A prototype of a system was designed and constructed that would detect the presence of an EMV, and turning the corresponding signal green, thus creating a green corridor for the EMV. For an emergency response vehicle, reaching its destination in the least amount of time is critical, and every second of journey time reduced makes a big difference. Hence by designing an IoT based system, which minimizes human-human and human-machine interaction, thus clearing

its path more efficiently, we believe that this is a step forward in saving thousands of lives every year, and at the same time, creating a smart city and hence a smart nation.

REFERENCES

[1] F. Andronicus and Maheswaran. “Intelligent Ambulance Detection System”.

[2] Dimitris K. Fragoulis and John N. Avaritosiotis. “A Siren Detection System based on Mechanical Resonant Filters”.

[3] . F. Meurcci, L. Pierucci, E. Del Re, L. Lastrucci and P.Desii. “A REAL-TIME SIREN DETECTOR TO IMPROVE SAFETY OF GUIDE IN TRAFFIC ENVIRONMENT”

[4] Takuya Miyazaki, Yuhki Kitazono and Manabu Shimakawa. “Ambulance Siren Detector using FFT on dsPIC”