Implementation of an Image Processing based Smart Parking System using Haar-Cascade Method

Imam Muhammad Hakim
Dept. of Electrical Engineering
Kalimantan Institute of Technology
Balikpapan, Indonesia
04161032@itk.ac.id

David Christover
Dept. of Electrical Engineering
Kalimantan Institute of Technology
Balikpapan, Indonesia
04151014@itk.ac.id

Adi Mahmud Jaya Marindra
Dept. of Electrical Engineering
Kalimantan Institute of Technology
Balikpapan, Indonesia
adi.marindra@itk.ac.id

***Abstract*—In highly populated cities, finding available car parking slots is time consuming and may cause severe traffic congestions at the parking entrance. Therefore, a smart parking system with automated car detection is required so that the car drivers would have minimum effort and time to access the available parking location. This paper presents an implementation of image processing based smart parking system using Haar-Cascade method. The aim of this project is to develop a smart parking system on a single-board computer and a camera installed at a parking area. The image processing for car detection is performed on Raspberry Pi interfaced with the Firebase cloud platform through an API. The system posts the data to the Firebase channel, enabling an Internet of Things (IoT) based parking system. A mobile application is developed for visualization and hence the data is accessible through the car driver's phone. The system was tested at a simple parking lot with a line of car slots. The experimental results with different condition of parking areas and different camera view angles are discussed in the paper.**

Keywords—Car detection, Haar-Cascade, image processing, internet of things, smart parking

I. INTRODUCTION

The world's urban population is expected to surpass six billion by 2045 [1]**.** Much of the expected urban growth will take place in countries of the developing regions, particularly Indonesia. As an impact of the growth of population, the number of land transportation vehicles in Indonesia has been increasing significantly. According to data from the Ministry of Industry in Indonesia, the number of land transportation vehicles grew 20% in 2012 [2]. This increase was due to population growth and positive economic conditions which spurred the increase in demand for land transportation. Along with the increase of land transportation, traffic jam and the number of parking spaces in many densely populated cities becomes more problematic. Particularly in public places, the limited parking slots lead car drivers to put a huge effort and time looking for locations to park their cars.

The limited parking area is still an unresolved problem. Researchers at UCLA reported that an average of 30% of drivers struggle to find slot parking of the day [3]. Optimizing parking lots is one solution to minimize this issue. Several studies in optimizing parking lots have been developed, such as using multiple sensors or using image processing [4]. In general, image processing can be used in optimization but is constrained by the problem of light, as well as the use of sensors installed in the parking area will easily detect non-vehicles but are counted as vehicle objects [5]. The accuracy in detecting empty parking areas is the main challenge for smart parking systems.

Based on the aforementioned problem and challenges, implementation of a smart parking system with experimental study is necessary. In this paper, we discuss the application of Haar-Cascade algorithm for detecting vehicles objects and empty spaces in the parking area using an application on smartphone. We also investigate experimentally the influence of parking distance and different camera view angles to the accuracy of the detection. Our framework envisions the use of camera and system will be sent data using the WiFi module on Raspberry to firebase cloud and then displayed in smartphone application that has been made. Delivery of parking area slots using applications on smartphones in real time is expected to facilitate reading and support digital systems that are currently growing rapidly such as Internet of Things (IoT).

II. LITERATURE REVIEW

A. Image Processing

Digital image processing is a manipulation and interpretation of images with the help of computers. In application, the image usually functions to improve the quality, make the process of withdrawing information or descriptions contained in the image and do compression or reduction of data. There are 2 types of images which are continuous and discrete. Continuous images are obtained from optical systems that accept analog signals such as humans and cameras, whereas discrete images are images produced from the digitization process of continuous images [6]**.**

B. Haar Cascade

Object recognition algorithms must use unique features for the object to be identified. In the human face, for example, a unique feature is that the area containing the eyes is darker than the area above the eye. Haar-Cascade algorithm has since become very popular in real time face detection systems. The general features of this approach can be effectively applied to detect other objects as well, including cars for a smart parking system [7]. As shown in Fig. 1 (a), this rectangle is a Haar feature that can be used to detect faces in images. Previous researchers describe the use of many simple features that can be classified into two groups of rectangles, three rectangles and rectangles. In the context of a car detection system, a similar Haar feature is shown in Fig. 2.

(a) (b)

Fig. 1. Illustration of Haar method for face detection: a) Single Haar b) 2 and 3 rectangle Haar applied to facial detection

(a) (b)

Fig. 2. Illustration of Haar method for car detection: a) Single Haar b) 2 and 3 rectangle Haar applied to car detection

C. *AdaBoost Learning Algorithm*

Our training data obtained from parking lots can be used with many classifiers. We use the algorithm proposed by previous researchers Viola and Jones because of its success in many detection tasks [8]**.** Viola and Jones use features such as Haar, integral images, Adaboost and cascade structures within their framework. Adaboost is used to select the most relevant classifiers. The main idea of the AdaBoost algorithm is to create a strong classifier as a linear combination of a simple weak classifier [9]**.** The single Haar feature can represent a simple weak classifier. The input for AdaBoost is a training set $S = (x_1, y_1), \ldots, (x_n, y_n)$, where x_1 represents the sample (in our case cars) and y_1 represents the possible evaluation (the positive samples are marked as 1, the negative samples has 0). This algorithm works in a repetitive way and changes the importance of data at each step. Finally, AdaBoost chooses efficient features based on the lowest errors and it creates strong classifiers using linear combinations of weak classifiers [10]. For the car detection process, we use the feature extraction scheme described above, implemented through OpenCV. By using these basic features, it is possible to create an effective classification cascade that can quickly detect vehicles with high accuracy.

D. *Firebase Cloud as an IoT platform*

Firebase is a combination of many Google services in the cloud, including instant messaging, real time databases, storage, and hosting. One of the main functions of the Internet of Things is data collection. Based on the Firebase Cloud, the controller connects directly to its database to read and write data. Firebase provides access to real-time databases on iOS, Android, JavaScript SDK, REST API, Admin SDK [11]. Android SDK and iOS are for developing mobile applications. The JavaScript SDK is for web applications. The REST API is a general network service, it can be used for general programming languages. Thus, the Firebase platform with Wi-Fi as a relatively fast and reliable wireless communication approach for data transfer.

E. *Hardware Platform*

The hardware model for this paper uses a camera and a Raspberry Pi 3 model B. The chosen minicomputer is the Raspberry Pi because the provided features are in accordance with the need for making this prototype. The Raspberry Pi 3 model B design is based around the Broadcom BCM2837 SoC (System-on-a-Chip), which has embedded QuadCore ARM processors with a clock speed of 900 MHz, Dual Band wireless LAN, GPU VideoCore IV, and 1 Gigabyte RAM. The camera used to carry out the image acquisition process is Pi Camera, which is a special camera designed for Raspberry Pi minicomputers. With a small size, the Raspberry Pi camera module can be used to take high definition quality images with 8 MP quality supporting 1080p 30fps video resolution, 720p 60 fps and VGA90, can work on all Raspberry Pi models connected to CSI ports [12].

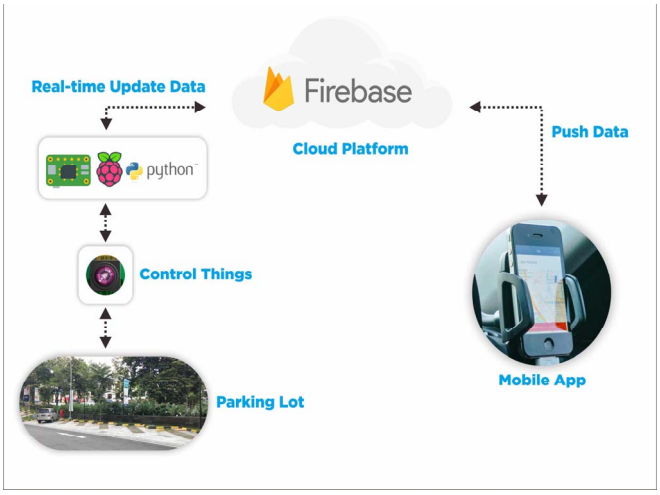

Fig. 3. Architecture of the implemented smart parking system

III. IMPLEMENTATION OF THE SMART PARKING SYSTEM

A. *System Architecture*

Architecture of the implemented smart parking area monitoring system is shown in Fig. 3. It consists of a Camera, a Raspberry Pi, a Firebase Database Cloud Computing platform, and a user side mobile application. The subsystems are interconnected with wired and wireless interfaces, as well as the internet connectivity. The structure of our system is explained in more detail as follows.

1) *Camera*

The system starts the whole process by taking real-time images via camera. Then, the images taken through the Pi camera color space will be processed. The image processing step in this system is the color filtering process. This process is carried out to take pixels in an image that has a certain color intensity value. By obtaining a particular color object, the next step is the thresholding process which aims to change the image with a degree of gray to a binary or black and white image so that it can be seen which area includes the object and background of the image clearly.

2) *Mini computer*

To facilitate image processing, it is necessary to have a minicomputer. The employed minicomputer is Raspberry Pi 3B which functions as a tool for processing images which have been taken by the camera. The Raspberry Pi is programmed with the algorithm described in the previous section. This minicomputer is also used to send the results of processing the data to the selected IoT cloud platform.

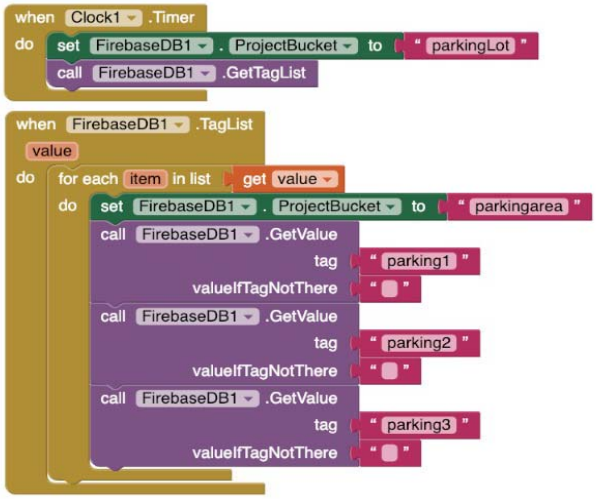

Fig. 4. Program blocks in the MIT Inventor

3) Firebase Database Cloud Platform

Firebase Database, which is owned by Google, supports a number of platforms and devices, such as iOS, Android, Linux (via Python), and Arduino firmware (via REST API) [13]. This compatibility is especially useful in time limited development cycles. First, it encapsulates development efforts into one centralized service instead of using different scattered services tied together. This reduces the development time and lets the engineer focus on other parts of the system being developed. Second, it allows the communication between different hardware systems with less effort and a shorter learning curve due to the availability of Software Development Kits (SDK) on different platforms. The SDK admin is used in the Node environment to implement the Python application. In this paper, firebase cloud serves as a storage place for data that has been sent, which is then displayed in the smartphone application created.

4) Mobile Development

Other components are visualization and data management. Visualization and data management in this system mainly through applications on smartphones. MIT App Inventor is an environment that leverages a blocks-based visual programming language to enable people to create mobile apps for Android devices [14]. An App Inventor project consists of a set of components and a set of program blocks that enable the functionality of these components. Components include items visible on the phone screen (e.g. buttons, text boxes, images, drawing canvas) as well as non-visible items (e.g. camera, database, speech recognizer, GPS location sensor). The app is programmed using Blockly, a visual blocks-based programming framework [15]. Fig. 4 shows the program blocks for an app to discourage texting while driving. When a text is received, a default message is sent back in response and the received text is read aloud.

A mobile platform is one of the parts which needs to be developed for this smart parking system. The mobile application is built using the MIT Inventor. It allows new users to computer programming to create applications for the Android operating system (OS) to make it easier for users to read empty parking lots in real time. The "data posting" in the MIT App Inventor is used as a command to post data manually from the input data to a cloud platform so called Antares.

Fig. 5. The open parking lot sample used in the experiment

IV. Experimental Results and Discussion

After developing the entire smart parking system, the results of parking spot and car detection via image processing are discussed in this section. We performed experimental investigations to show the interactions between each subsystem explained in the earlier section. The experiments were conducted at the parking area of Living Plaza Shopping Mall in Balikpapan city. Two different experimental conditions considering camera view angle and distance between cars are the main objects to analyze in this section.

A. Detection of Parking Spots

To detect objects using a camera, the first step to take is creating frames from the captured parking lot. Fig. 5 depicts the open parking lot sample used in the experiment. There are 8 parking slots marked with A1 to A8. The image also includes a number of small red colored squares as the indicators whether any object has been parked at those specified spots. This pre-processing step is useful for triggering the next action in detecting if a car exists at the spot.

B. Detection of Cars

After determining the parking spots on the camera view, the next step is to detect the object at those particular spots. When some object exists at the pre-defined spots, the system will start detecting whether it is a car. The OpenCV library is used for linking the process of using the Viola-Jones algorithm for car detection. The library files should be extracted and converted to xml files to be used in Python. The function for conversion to xml file is already available on the framework of Viola-Jones. All components for car detection such as the framework of Viola-Jones and the OpenCV library need to be inserted to a compiler to develop a car detection functionality. The Haar-Cascade classifier is used to remove objects other than cars using a strong classifier which has been trained by the AdaBoost in many classification levels. In this work, the training data is formatted with 100x100 Pixels for both positive images and negative images. Then, the training results are cropped into 80x70 Pixels because the width of cars is generally longer than the height of cars. After resizing and cropping the

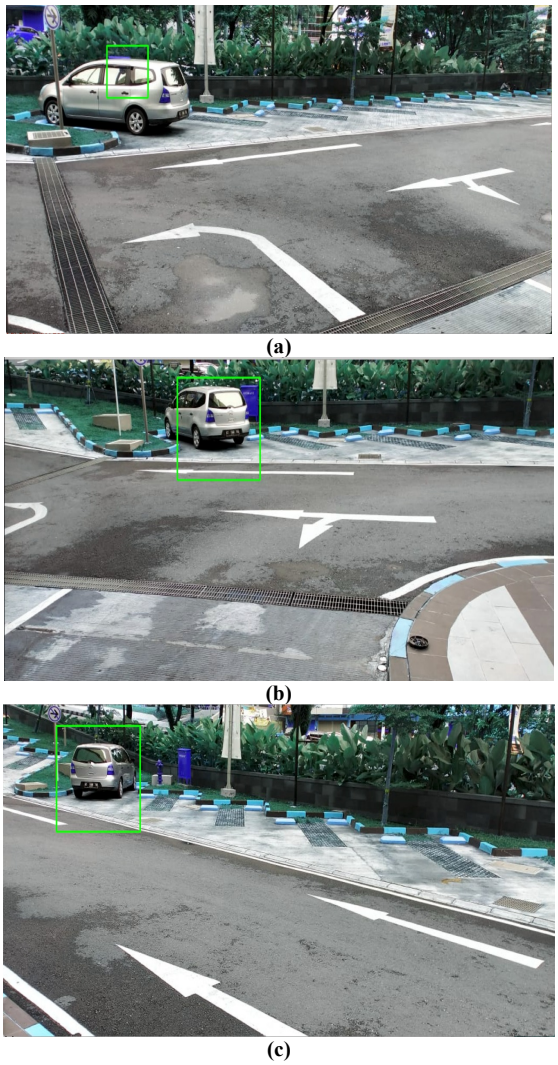

Fig. 6. A single car detection result with the camera positioned at high elevation and different view angles: (a) view from left, (b) view from center, dan (c) view from right

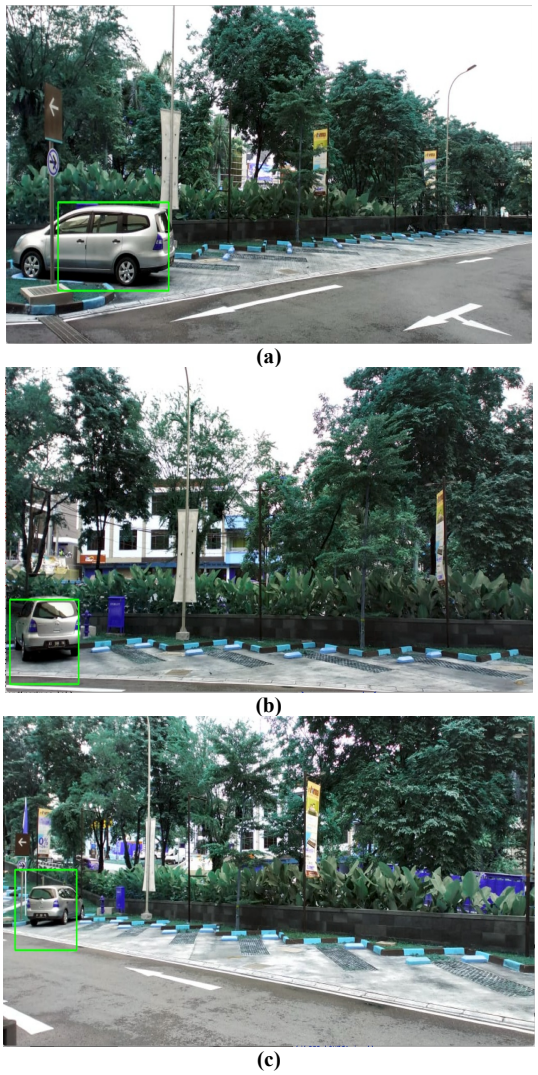

Fig. 7. A single car detection result with the camera positioned at low elevation and different view angles: (a) view from left, (b) view from center, dan (c) view from right

parking area as shown in Fig. 6, thresholding process is performed to adjust the color of the image. An image of the parking area is taken for the reference image. Later on, the image is taken in real-time for car detection. In this work, two variable conditions are analysed. First, we analyse how the placement of camera, which subsequently change the view angle influence the detection accuracy. Second, we also analyse how the distance of the parked cars affect the detection accuracy.

It can be seen from Fig.7 and Fig. 8 that mounting the camera at a high elevation makes the street captured more dominantly, while mounting the camera at a low elevation causes unwanted objects such as buildings, windows, and advertising banners included in the image. Before experiments, we assumed that the unwanted objects in the camera view would affect the accuracy as other objects might be detected as cars. In some parking lots, particularly at shopping center areas, unwanted objects cannot be avoided. However, our experiments show that the system can detect the number of cars accurately regardless the different camera mounting elevation and view angles of the camera. A single car can be easily detected from the left side, from center, and from the right side of the car.

Fig. 8. Multiple car detection result in which the cars are seen as having different gap distances. The shadows under the cars also cause detection error.

Fig. 8 illustrates a case when multiple cars exist at the parking lots and the camera sees the cars as having different gap distances. The different gap distance is actually caused by the perspective view of the camera, which commonly happens in practical situation. From the image processing, it can be inferred that the system can detect cars accurately if the distance between cars is not too close to each other. When the distance between cars is seen too close, the system may detect multiple cars as one. Moreover, shadows of the

TABLE I. EXPERIMENTAL RESULTS FOR DIFFERENT VIEW ANGLES AND DISTANCE BETWEEN CARS

No	Variable	Number of car parked	Number of car detected	Accuracy
1	Single car viewed from top left	1	1	100%
2	Single car viewed from top center	1	1	100%
3	Single car viewed from top right	1	1	100%
4	Single car viewed from bottom left	1	1	100%
5	Single car viewed from bottom center	1	1	100%
6	Single car viewed from bottom right	1	1	100%
7	Multiple cars with different gap distance and shadows under the cars	8	10	80%

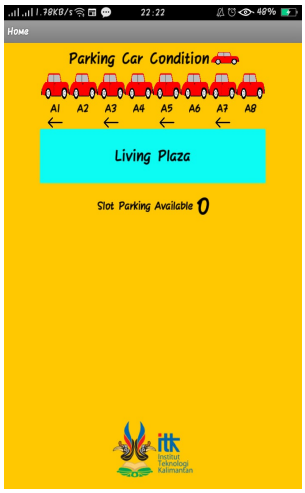

Fig. 9. User interface on the mobile application

cars around the tyres could be detected as cars and cause some duplication in the car detection. It is because the shadows formed below the cars have similar black and white color pattern as the car's windshield. The experimental results are also summarized in Table I. It is worth noting that the gap distance between cars as seen by the camera is crucial because it reduces the detection accuracy in the application. This issue may distort the occupied and the available parking slots for the smart parking system. It is recommended that the camera is placed at the center back of the parking slots line to avoid perspective view, which leads to the different car gap distance. The camera should be also installed at slightly high elevation to avoid shadows formed below the cars.

C. *Data Transfer to the IoT Platform*

The data transfer from the Firebase to the mobile application has been done successfully via API keys. The process in the mobile apps starts from the request from the app to the Firebase server via internet connection. Afterwards, the API responds the request and forwarded the message to the Firebase Cloud Messaging server. Subsequently, the Firebase Cloud Messaging server replies the message to the mobile app user so that the app get the requested data in the form of string which is visualized on the app. Fig. 9 exhibits the user interface of the app with the information of car parking condition. This condition is obtained when the parking lot is fully occupied by multiple cars as in the case shown in Fig. 8. In this case, the camera and Raspberry Pi on the site process and send the car detection result to the Firebase cloud server. The data in Firebase server can be accessed as requested automatically by the mobile application. When the parking lot is full, the parking slots A1 to A8 are depicted with a picture of cars meaning that the corresponding parking slots are being occupied. The detection result suits the actual condition in the parking area.

V. CONCLUSION

A smart parking system has been implemented and discussed in this paper. The system is based on image processing using Haar-Cascade method and supports the IoT concept by making the data available on internet using Firebase platform and also accessible through a mobile application. The main findings are related to the car detection using Haar-Cascade method. It is found that the single car detection with different view angles using Haar Cascade method does not influence the detection capability. The car is always detectable from any view angle. However, when multiple cars are parked and the gap distance between cars is seen too close by the camera, the detection becomes less accurate. Two cars may be detected as one car. Furthermore, the shadows under the cars may cause detection errors as the color pattern is similar to the car's windshield.

For future works, the data transfer from Firebase to the mobile application using the MIT Inventor could be developed further supporting more user-friendly interface and more rich information. Most importantly, the car detection method should be improved to avoid detection errors associated with the gap between cars and the shadows formed around the cars.

REFERENCES

[1] "World's population increasingly urban with more than half living in urban areas | UN DESA Department of Economic and Social Affairs," *United Nations*. [Online]. Available: http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html. [Accessed: 30-Jan-2019].

[2] "Kemenperin: Kendaraan Angkutan Darat Tumbuh 20%," *Kementerian Perindustrian*. [Online]. Available: http://www.kemenperin.go.id/artikel/4649/Kendaraan-Angkutan-Darat-Tumbuh-20. [Accessed: 30-Jan-2019].

[3] Donald. C. Shoup, "The High Cost of Free Parking", Chicago:Planners Press, American Planning Association, 2005.

[4] X. Xiang, N. Lv, M. Zhai and A. El Saddik, "Real-Time Parking Occupancy Detection for Gas Stations Based on Haar-AdaBoosting and CNN", in IEEE Sensors Journal, vol. 17, no. 19, pp. 6360-6367, Oct.1, 1 2017.

[5] ChihPing Hsu, Toshimitsu Tanaka and Noboru Sugie. "Locating Vehicles in Parking Lot by Image Processing". IAPR Workshop on Machine Vision Applications. 2002.

[6] Leonard Satrio Tegar, Jana Utama. " Designed Build Information System in UNIKOM Four Wheeled Parking Lot based on Image Processing". Telekontran Vol 4 No 1 April. 2016.

[7] Praveen Meduri, and Eric Telles "A Haar-Cascade classifier based Smart Parking System. International Conference. IP, Comp. Vision, and Pattern Recognition. 2018.

[8] P. Viola and M. Jones, “RaPid object detection using a boosted cascade of simple features”, Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR pp. I-511-I-518 vol.1. 2001.

[9] Y. Freund and R. E. SchaPire. “A Short Introduction to Boosting”. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September 1999.

[10] Radovan Fusek, Karel Mozdren, Milan Surkala and Eduard Sojka. “AdaBoost for Parking Lot Occupation Detection”. Ostrava university. Unpublished.

[11] Wu-Jeng Li, Chiaming Yen, You-Sheng Lin, Shu-Chu Tung, and ShihMiao Huang, “JustIoT internet of things based on the firebase real-time database”. IEEE journal. 2018.

[12] Monica Chillaron, Larisa Dunai, Guillermo Peris Fajarnes, and Ismael Lengua. “ Face Detection and Recognition Application for Android”. IECON. 2015.

[13] Divyesh Zanzmeriya, and Ankita Panara. “Implementation of Industrial Automation Systems using Raspberry Pi by IOT with Firebase”. International Research Journal of Engineering and Technology. Vol 5, issue 5. 2018.

[14] Xie, Benjamin, Isra Shabir, and Hal Abelson. "Measuring the Usability and Capability of App Inventor to Create Mobile Applications." 2015 ACM SIGPLAN Conference on Systems, Programming, Languages and Applications: Software for Humanity SPLASH. October 2015.

[15] “Blockly | Google Developers,” *Google*. [Online]. Available: https://developers.google.com/blockly/. [Accessed: 30-Jan-2019].