2019 International Conference on Advancements in Computing (ICAC)
December 5-6, 2019. Malabe, Sri Lanka

Design and Implementation of an Autonomous Indoor Surveillance Robot based on Raspberry Pi

Tharindu Dharmasena, Pradeep Abeygunawardhana
Faculty of Computing, Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
tharindu.d@sliit.lk, pradeep.a@sliit.lk

***Abstract*—In recent years robotics has influenced many fields including the security and surveillance domain. Due to convenience and flexibility more and more security robots tend to be deployed in place of humans for routine activities such as area sweeps. While there are many kinds of research have been done regarding this concern, many of the solutions cost more due to their implementation complexity while low-cost implementations are only capable of doing simple activities such as following given local coordinates. This paper describes an autonomous surveillance robot that is being developed while keeping the development costs as low as possible and is capable of performing routine patrols autonomously in indoor environments and detect anomalies around it such as temperature fluctuations, unauthorized personals and report them back to a central computer. This robot can be controlled remotely by security personals to facilitate manual inspections. Due to the development architecture of the robot, more software-based features can be added easily.**

***Index Terms*—Raspberry-Pi, Surveillance, Face Recognition, Autonomous-Navigation**

I. Introduction

Security surveillance is a growing concern among the business community and homeowners. In 2011 homeowners in the USA alone have spent about $20.64 billion on home security systems[1]. With the recent developments in the field of Robotics, autonomous surveillance robots have begun assisting security personals in surveillance operations. While both aerial and ground robots assist in most of these activities, ground robots are heavily used to do surveillance activities because ground robots are cheaper to implement and easy to deploy when compared with aerial robots. At the same time, these autonomous guard robots can be used as a remote eye by transmitting live video feeds back to security personals.

The first surveillance robot which was used for security purpose was Mobile Detection Assessment and Response System (MDARS) Everett, H. & Gage, D.W., 1999 [2][3]. Since then the development of mobile surveillance robots increased rapidly and some of these robots are used in military operations as well. These robots can be remotely manipulated or can be assigned specific tasks like navigating for a specific point autonomously. In a research carried out by Yoichi Shimosasa and team they came up with a system which combined security surveillance system and service system. The robot was capable of guiding visitors to their destinations in day time and patrolling at night[4]. Another team developed a group of robots that are distributed among different floors of the building and can report abnormal events[5]. In 2011 Hou-Tsan Lee and team built a robot with a wide range camera attached to it to replace the traditional fixed position security cameras which can autonomously navigate with the help RFID tags placed across the area[6].

Several robotic surveillance systems mentioned above are developed using high-performance onboard computers and actuators thus utilizes more power while the robot is operating. This reduces the battery life of the robot which in turn reduces the operating time of the robot on the field. Some systems which are developed orienting low development costs have only basic capabilities that are required to operate as a surveillance robot.

The mobile surveillance robot described in this paper is developed using a Raspberry-Pi development board running Raspbian operating system and ROS (Robot Operating System) framework. It will autonomously patrol the specified routes on a schedule while collecting sensor data and send them to a central server for further processing. The server acts as the ROS master and the mobile robot acts as a ROS slave. The development process of this robot focuses on keeping the development costs as low as possible while integrating high-level functions such as human recognition, night vision capabilities, 2D indoor mapping and autonomous navigation.

II. Design of the Robot

A. Mechanical Structure and Sensor Placement

The robot is propelled by 2 DC geared motors which operate at 12V with 5000 rounds per minute. The rotational power from the motors is used to propel the robot using a traction belt which runs through several free rotating wheels and makes it able to navigate in rough terrains and to increase the traction in indoor environments. The wheels are spring-loaded so that the chassis can absorb the shocks so it reduces the chance of changing its heading abruptly while traveling at higher speeds. To collect the information on the environment, the robot is equipped with several sensors; lidar, gyro and accelerometer (MPU6050), magnetometer (HMC5883L), near- infrared camera (RaspberryPi NoIR camera), temperature and humidity sensor (DHT22)

978-1-7281-4170-1/19/$31.00 ©2019 IEEE

Some of these sensors' information is used to localize the robot in the environment to facilitate autonomous navigation in the operating field. A separate USB WiFi adapter is plugged into the Raspberry-Pi in addition to the built-in WiFi module which will be used to establish the connectivity between robot and the ROS master.

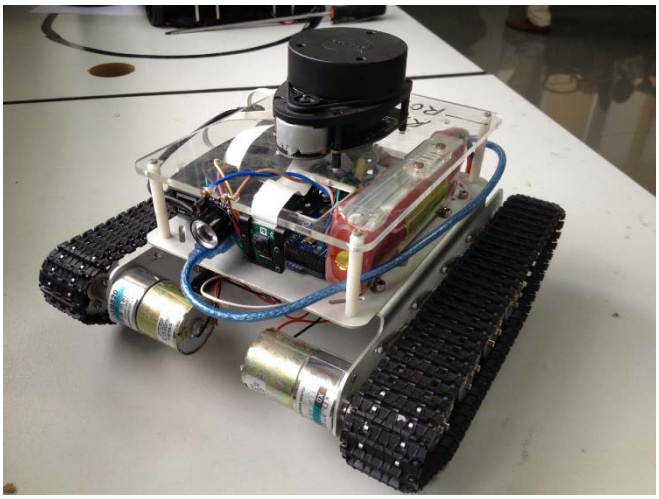

Fig. 1. Current physical appearance

Motors of the robot are powered from a 12V LiPo battery and controlled by a L298N motor controller which is connected to the RaspberryPi via GPIO(General Purpose Input Output) pins. Since the localization process mainly depends on lidar data, the motors operate at a lower speed by using a 70% duty cycle to avoid localization problems that could occur if moved at higher speeds. The gyro and accelerometer sensor, temperature sensor, and magnetometer are connected to an Arduino-mega development board and the data from these sensors are collected on the Arduino board itself. Odometry data is calculated inside the Arduino using wheel encoder readings along with the data collected from the accelerometer and magnetometer. These collected sensor data and calculated information are then published into relevant ROS topics using ROS Serial library for Arduino and then collected using ROS serial node running in the Raspberry-Pi.

A 6000mAh power bank is used to powers up the RaspberryPi. An Arduino mega board which handles all the sensors and motors is connected to the Raspberry-Pi via a USB port which provide power to the Arduino while facilitating data communication between the RaspberryPi and the Arduino.

The robot is equipped with a NoIR RaspberryPi camera module which is capable of seeing the surroundings in a dark environment with the help of a separate Infrared light to illuminate the area. The infrared illuminator attached to the robot is a 3W bulb that is capable of operating at voltages between 3.3V to 5V. The NoIR camera connects with the RaspberryPi via CSI slot available on the board like an ordinary PiCam would do. Even though the NoIR camera takes images in grayscale mode in dark environments, the same PiCam is capable of taking color images in normal daylight conditions as well.

A 'RPlidar-A1' lidar (Light Detection and Ranging) is placed on top of the robot which eliminates the possibility of parts of the robot being in the vision range of the lidar. The data form the Lidar is used to create a map of the environment and also used in autonomous navigation. The Lidar is capable of taking distance measurements up to 12 meters with a maximum frequency of 5.5Hz.

The RaspberryPi, Arduino board, LiPo battery, power bank and other small accessories are placed in the area between the chassis and the acrylic sheet. The motor controller of the robot is placed underneath the chassis.

B. Software Architecture

Once the robot is switched ON it will connect with the predefined WiFi network available on the floor to connect with the ROS master which is running on a remote server. Inside the server, all the high-level functions of the robot are running as ROS nodes. These nodes will subscribe to the required topics to get the data that was collected by the robot and then published in to. These nodes are implemented on the server-side because more processing power is available in the server. The software of the robot is implemented on ROS (Robot Operating System) framework using a combination of Python and C++ languages. The sensors mounted on the robot are read by the Arduino board and the sensor data are published on to their respective topics via rosserial library. These data on topics are then accessed by other ROS nodes where the data is needed in the process of decision making. The Raspberry- Pi creates a password-protected hotspot upon the boot-up which facilitates the authorized users to connect with the robot directly. In the current development stage, the ability to initiate simultaneous connections through the builtin WiFi module and the USB WiFi module is not implemented. Camera feed from the PiCam is streaming continuously and it is accessible from any computer which is connected to the same security network. Images from the same video feed is captured by the relevant ROS nodes to detect and recognize human faces in the frames. To do the face recognition OpenCV libraries for python and a pre-trained CNN which is available in python library 'face recognition' was used[7]. Separate ROS nodes have been implemented for specific actions the robot has to do such as navigating, combining sensor readings, receiving navigation commands from users, human detection and face identification. To facilitate the serial communication between the Arduino board and the ROS framework in RaspberryPi via USB port, rosserial package was used in the RaspberryPi and rosserial library in the Arduino code.

III. FUNCTIONALITIES OF THE ROBOT

A. Communication

Since the RaspberryPi on mobile robot doesn't have enough processing power to do all the data collection and processing, the robot itself acts as a mobile data collection node and sends collected data to a central server where actual data processing and analysis takes place. The communication between the

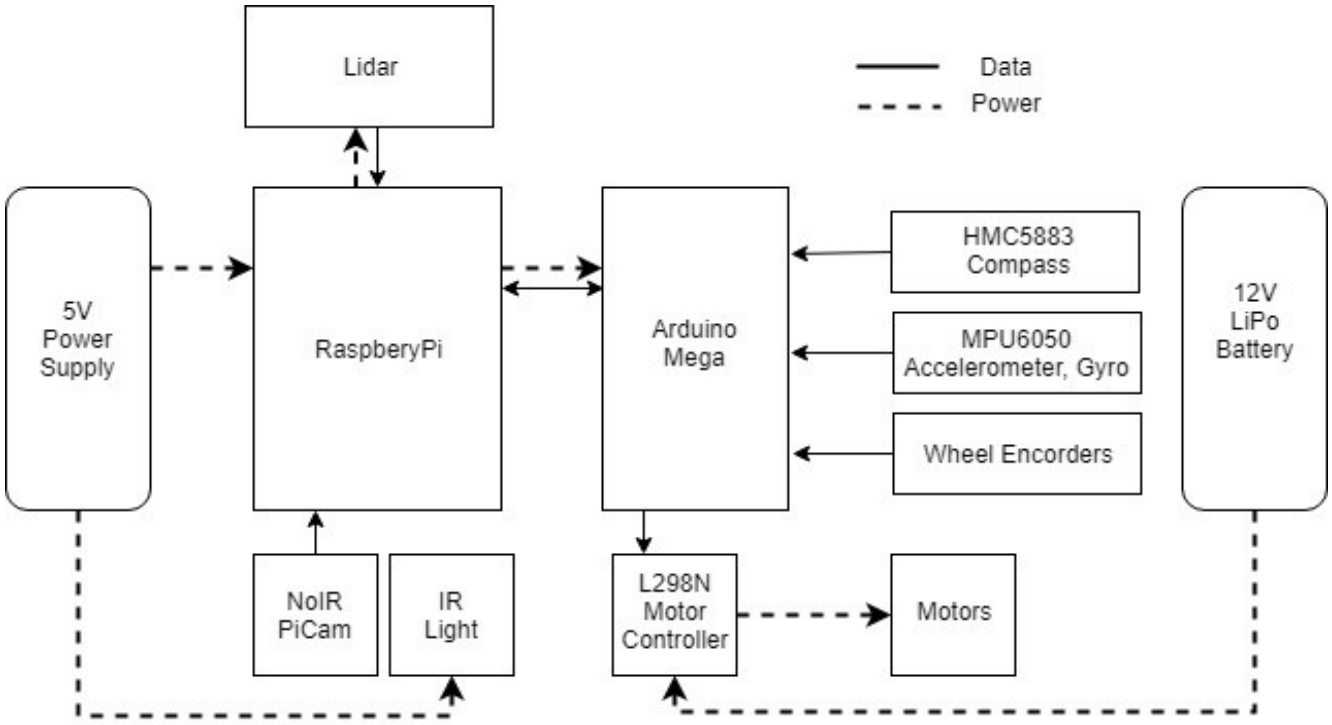

Fig. 2. Hardware layout of the robot

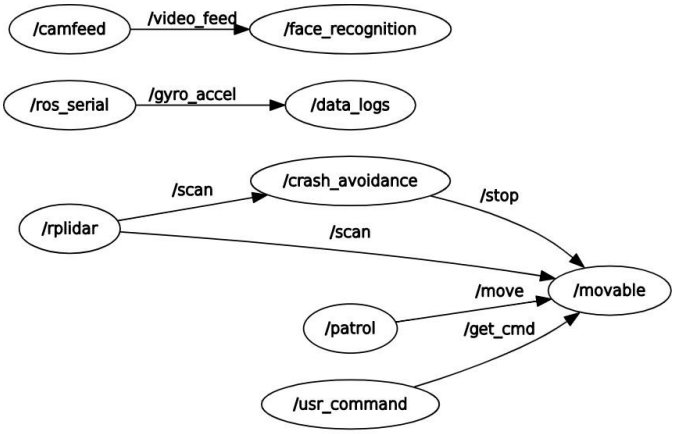

Fig. 3. rqt graph of the ROS nodes

Fig. 4. Generated map of the floor

robot and the central server is established via Wi-Fi communication. To make reconfigurations and modifications to the robots program easier, upon the boot-up, the robot creates its own hotspot which can be used to connect with the robot. In the development stage, ROS master is running on a laptop and the laptop has to be connected with the WiFi hotspot created by the Raspberry-Pi.

B. Indoor Mapping

To navigate autonomously, a robot has to have a map or has to create a map of its environment so that it can localize itself in the environment. To create the map of the environment using only the Lidar data 'Hector SLAM' (Simultaneous Localization and Mapping) was used since it is easier and quicker to implement compared to other localizing techniques such as 'Gmapping' or 'Cartographer' which require additional odometry information and sensor data[8]. Even

though the processing power of RaspberryPi is not sufficient for complex computations like map generation it is used to collect the lidar data and publish them into '/scan' topic. This lidar data are then captured by the hector slam nodes running in the ROS master to generate the map of the environment. In addition to that IMU data also gets collected using Arduino which can be later used to refine and improve the accuracy of the localization process. The mapping process has to be carried out in a semi- autonomous mode. A user has to connect to the robot and then have to guide the robot manually until the whole area is completely mapped where the robot is supposed to operate. In this mode, a separate ROS node is running inside the robots system which stops the movement, if the robot itself is going to collide with an obstacle.

C. Navigation

The 'moveBase' ROS node is responsible for controlling the motors of the robot when it needs to move. This node is subscribed to 3 topics to receive navigation commands from the user, collision avoidance node and patrol node. In the map generation process, the user has to manually guide the robot. In that process, if the robot gets too closer to an obstacle, the node which is running to avoid collisions sends stop command to 'moveBase'. But the user can send commands despite the obstacles in front of it but shortly collision avoidance will again stop the robot from moving in that direction. 'patrol' node is responsible for sending navigation messages to move between the given coordinates. The priority to messages 'moveBase' node receives are assigned in the following order; user messages, collision avoidance messages, patrol messages.

D. Face Recognition

To do the facial recognition OpenCV, Python and a deep learning method called 'deep metric learning' were used. Traditionally neural networks used in face detection or recognition usually takes an image as an input and outputs a draw box coordinates or a label for detected or recognized faces but the neural network used in here which is available in the 'dlib' library is capable of detecting a face in an image or video stream and generating 128-d embedding (quantifications) for that face. This output is then matched with a 128-d embedding which we have generated previously for known faces to identify the person appearing in the image[7][9]. This capability is used by the robot while doing its routine patrol to detect unauthorized people that are captured in the video feed.

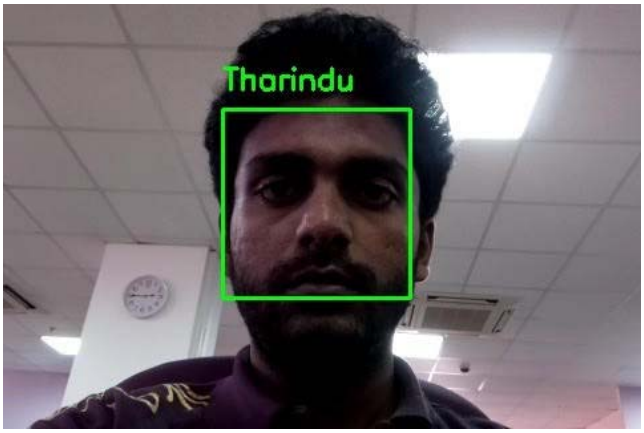

Fig. 5. Saved result from face detection node

IV. Results and Discussions

Since ROS framework is running along with several ROS nodes in the RaspberryPi, the CPU utilization reaches its maximum which results in heating the board very quickly.

To maintain the operating temperature of the CPU in a safe level a heat sink was attached on top of the CPU and a cooling fan was fixed to actively cool down the Heatsink. Removing ROS nodes that contains high-level functions such as facial recognition from the robot's onboard computer and moving them to the central server can greatly reduce the processing load for the robot.

The RaspberryPi 3 model B+ have 5V GPIO pins which are capable of powering up the IR illuminator. But since the light alone required about 600mA to operate at its maximum power, it was powered directly from the power bank. This procedure avoided the unnecessary heat build-up in the power lines of the RaspberryPi and reduces the possibility of the development board running in low power which could damage the board itself and other components connected to it. When the IR illuminator is operating at its maximum power the circuit board which holds the light itself also started to heat up. To remove this excess heat, another small heat-sink was attached to the back of the circuit board of the light to increase the rate of heat dissipation.

The images used for face detection are not affected by noise in normal lighting conditions but in dark environments, the near-infrared images become noisy when the distance between the camera and the person increases. The current configuration can see objects available within 1m range clearly but when the distance between the robot and the object increases, the image becomes darker and less clear. To overcome this issue, multiple IR illuminators can be used to illuminate the environment more effectively or high powered IR illuminators can be used which can drain the battery faster.

An Arduino-Mega was chosen to included in the robot as it has a larger memory and can be used to execute larger code segments along with several other helping libraries included without running out of memory. Arduino-Nano's memory was sufficient to hold the 'rosserial' library but was not enough to store the other libraries which were used to get the reading from the other sensors of the robot.

V. Conclusion

With the advancements in robotics, robotic surveillance domain is becoming popular. The main drawback of a robotic system is the cost it takes to develop and implement. This paper described a surveillance robot that was implemented

using low-cost development boards and sensors which help to reduce the final development cost of the system. Since all the software packages used in the development process are open source, no additional charges are added to the development process and can be changed to cater to the requirements of the system. Even though the development cost is kept as low as possible, the system is capable of navigating autonomously while engaged in routine patrols, recognizing intruders via facial recognition and can act as a mobile camera that can be used by security personals to check the environment even in dark conditions.

REFERENCES

[1] J. Garcia, A. Alsuwaylih, and S. Tosunoglu, "Security Patrolling Autonomous Robot," in *Florida Conference on Recent Advances in Robotics*, 2015, pp. 14–15.

[2] B. Shoop, M. Johnston, R. Goehring, J. Moneyhun, and B. Skibba, "Mobile detection assessment and response systems (MDARS): a force protection physical security operational success," 2006, p. 62301Y, doi: 10.1117/12.665939.

[3] K. Pooventhan, R. Achuthaperumal, S. Kowshik, and C. . Manoj Balajee, "Surveillance Robot Using Multi Sensor Netwok," *Int. J. Innov. Res. Electr. Electron. Instrum. Control Eng.*, vol. 3, no. 2, pp. 1131–115, 2015.

[4] Y. Shimosasa *et al.*, "Some results of the test operation of a security service system with autonomous guard robot," in *IECON Proceedings (Industrial Electronics Conference)*, 2000, vol. 1, pp. 405–409, doi: 10.1109/IECON.2000.973184.

[5] S. Chia, J. Guo, B. Li, and K. Su, "Team Mobile Robots Based Intelligent Security System," *Appl. Math. Inf. Sci.*, vol. 7, no. 2, pp. 435–440, 2013.

[6] H.-T. Lee, W.-C. Lin, and C.-H. Huang, "Indoor Surveillance Security Robot with a Self-Propelled Patrolling Vehicle," *J. Robot.*, vol. 2011, pp. 1–9, 2011, doi: 10.1155/2011/197105.

[7] A. Rosebrock, "Face recognition with OpenCV, Python, and deep learning - PyImageSearch," 2018. [Online]. Available: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/. [Accessed: 21-Jul-2019].

[8] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," in *2018 International Conference on Intelligent Systems (IS)*, 2018, pp. 400–407, doi: 10.1109/IS.2018.8710464.

[9] D. King, "dlib C++ Library: High Quality Face Recognition with Deep Metric Learning," 2017. [Online]. Available: http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html. [Accessed: 24-Aug-2019].