2015

MRS: an Autonomous and Remote-Controlled Robotics Platform for STEM Education

Timothy Locke
Middle Georgia State College, timothy.locke@mga.edu

Peter Colon
Middle Georgia State College, peter.colon@mga.edu

Myungjae Kwak
Middle Georgia State University, myungjae.kwak@mga.edu

MRS: an Autonomous and Remote-Controlled Robotics Platform for STEM Education

Timothy Locke
Middle Georgia State College
timothy.locke@mga.edu

Peter Colon
Middle Georgia State College
peter.colon@mga.edu

Myungjae Kwak
Middle Georgia State College
myungjae.kwak@mga.edu

ABSTRACT

It has been known that robotics can engage students in STEM fields by providing interactive, hands-on, cross-disciplinary learning experiences. This paper represents a multipurpose educational robotics platform designed and implemented by a group of IT students by combining various currently available robot technologies. Its main functions include differential drive, autonomous obstacle avoidance, torso and arms movement, real-time video streaming, and remote control. The robotics platform was developed to engage students at middle school, high school, and college levels in science, engineering, and computer programming. It will be experimented and further enhanced by students in robot programming course at a college.

Keywords

Robotics, STEM education, Arduino, Raspberry Pi, Robot Programming

INTRODUCTION

A fast evolving, technology-driven modern society increasingly needs a well-trained science, technology, engineering and mathematics (STEM) workforce equipped with innovative problem solving and higher order thinking skills. Robotics programs are excellent in fulfilling the needs. Robotics engages students in STEM areas by providing interactive, hands-on, minds-on, cross-disciplinary learning experiences (Barker & Ansorge, 2006; Habib, 2012). Studies show that flexible curriculum can accommodate changing educational needs by using available resources and that informal science experiences including robotics programs have a strong influence on the STEM career choices of students (Habib, 2012; Connaughton & Modlin, 2009).

This paper presents the MGARL Robotics System (MRS), a robotics framework designed to engage students at middle school, high school, and college levels in science, engineering, and computer programming principles and developed by a group of IT undergraduate students. The objectives of the project is to develop an autonomous robot as an active learning tool for college students and an educational platform with which students can experiment so that they can be more engaged in STEM education. The next phase of the project is to conduct a series of experiments in a robot programming course of a college to examine whether the developed robot platform really serves for the intended purpose.

RELATED WORK

Many studies have been done for using robotics for undergraduate science education. Ruzzenente et al. examined the tools that are available to incorporate robotics into their science and engineering curriculum (Ruzzenente, Koo, Nielsen, Grespan, & Fiorini, 2012). Dodds et al. researched the robotics resources available to artificial intelligence courses (Dodds, Greenwald, Howard, Tejada, & Weinberg, 2006). Cielniak et al. integrated robotics education into an undergraduate Computer Science curriculum including the programming of vision-guided mobile robots in an open-ended assignment (Cielniak, Bellotto, & Duckett, 2013). Mobile computing can also be employed in robot education (Chattopadhyay & Sellman, 2013; Kurkovsky, 2014). Some work have focused on integrating a variety of sensors to the robot control architecture, most notably the integration of digital camera to enable students to develop computer vision algorithms and test wireless communication of

video streams captured by the digital camera (Saad, 2004). Remote control has been an interesting area. Educational robot researchers have investigated web based remote control (Yu, Tsui, Zhou, & Hu, 2001), which can be considered to integrated to web programming courses, and ecological interface for navigation (Nielsen, Goodrich, & Ricks, 2007).

MGARL ROBOTICS SYSTEM

Overall Architecture

Many robotics kits and electronics have been developed for both personal and academic purposes. In designing our robot system, our purpose was to combine various currently available technologies to create a multipurpose robotics platform with which students could experiment. Figure 1 shows the overall system architecture of the MGARL robotics platform. It was designed to have differential drive, autonomous obstacle avoidance, torso and arms movement, real-time video streaming, and remote control.

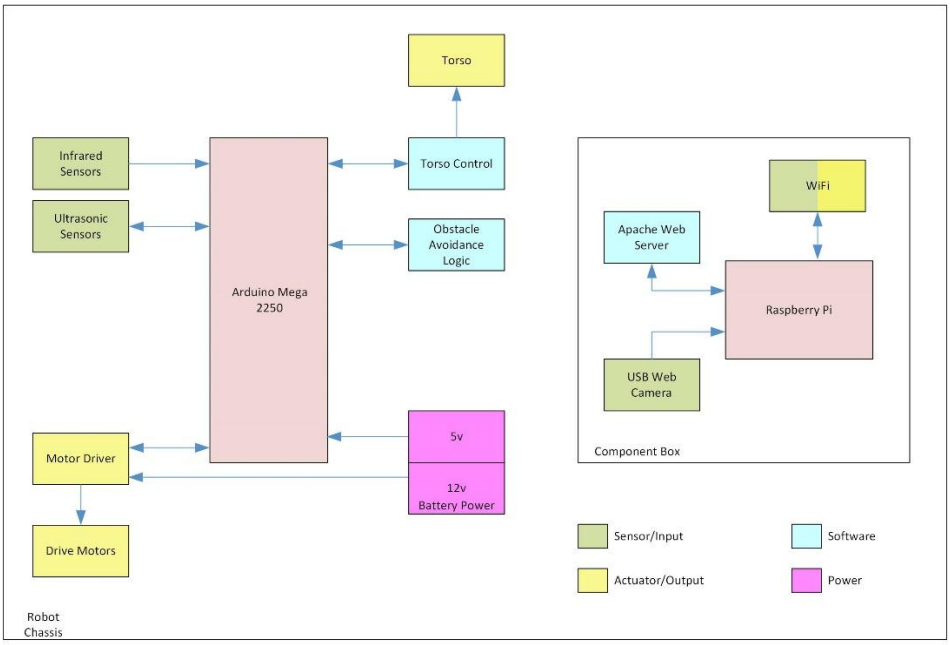

Figure 1. Robot System Architecture

Implementation

Differential Drive

One requirement of this project was the robot needed to have a differential drive system. Implementing a differential drive requires a minimum of two motors, which can run forward and reverse directions. A Pololu Qik motor controller (Pololu, 2015) was selected to drive the motors since it could handle the stall current of the used motors. A differential drive system works by driving each motor separately. To move forward, one motor runs forward and the other motor runs in reverse at the same speeds. If the robot needs to turn while moving, one motor will turn slower than the other. If the robot needs to turn in place, both motors will turn in the same direction at the same speed. This may be difficult to visualize, but since each motor is on the opposite side, the rotation of the shaft has to be opposite of the other to achieve a non-turning direction.

States and Delays

The foundational code for the robot was written by using a multi-sensor module from NewPing Library for Arduino (Eckel, 2014). The timing in the code was critical and any *delay* functions called within the main loop could cause problems. Any tasks requiring real time capture of data such as obstacle detection and avoidance algorithms cannot have the *delay* function, which could cause the operation of the controller to halt. In case the controller stops working, no avoidance maneuvers can be initiated and a collision is likely to happen. Instead of using the Arduino *delay* function, a timing routine coupled with a

Boolean state variable was adopted and it worked nicely. In the timing routine, states (or flags) allow the controller to evaluate and process events immediately. There are four states implemented with this robot: *stopped*, *moving*, *turning*, and *backward*. If the robot is commanded to move (forward in this case), it enters a state of *moving* which is true. A timer then is *started* and at a predetermined time a sensor is checked or a command is updated. The controller is still running and no event is missed.

Obstacle Avoidance

Ultrasonic and infrared sensors were used in the robot platform. The ultrasonic sensors work by triggering a high frequency sound, which travels at the speed of sound, is reflected off an object, and returns to the sensor as an echo. The round trip time is determined and distance is calculated. The Infrared sensors emit an infrared beam of light, which is reflected off the object and returns to the sensor where the round trip time is determined. The beam travels at the speed of light, so the round trip times are much smaller. How the sensors are implemented is different. The Ultrasonic sensor uses digital pins and code triggers the sensor and waits for an echo. The sensors are sampled every 33 milliseconds. The Infrared sensor uses an analog pin and is continuously sampled.

The original requirement for this robot was to use only ultrasonic sensors to create a three hundred and sixty degree force field boundary for object detection. During extensive testing, this implementation did not work as expected. Many objects were missed due to the different heights. In addition, it was hard to return a sonic echo from an angled surface such as a wall corner. Infrared sensors were added to the bottom of the robot and pointed to cross just in front of the opposite track. Two were placed which provided coverage for the front lower area of the vehicle. Software filtering limited the detection distance to a couple of centimeters on each side of the track.

The "or" logical operator was used to determine if a sensor is detecting an object within critical distances. A function was created to return a number for the specific ultrasonic sensor, which is detecting the obstacle. To do so, each sensor was assigned a number; the left sensor is number 1, the center is number 2, and the right sensor is number 3. In addition, a zero means no sensor is detecting an object. The infrared sensors are assigned a specific name and since they are continuously sampled, they do not need to return a number. They are checked, and returning a true value means that an object is detected.

Once sensors detect an object, software algorithms determine if it is an obstacle. Since the speed of the robot is not that fast, an object detected within 20 centimeters or less can be processed and an avoidance maneuver initiated. Distance from the infrared sensors is filtered to only consider an object at or closer than 14 centimeters to be an obstacle. If an obstacle is detected, for example, from the left ultrasonic sensor, the state is changed from moving to turning, the timer is started and a right turn is initiated. Once the timer has elapsed, the sensor is checked and a decision is made to continue the turn while the controller is still processing other events. Since the elapsed time is very short (approximately 36 milliseconds), the robot still can move forward while turning. If an obstacle is detected using the infrared sensors, the robot will stop, backup a short distance and initiate a turn depending upon which sensor detected the object.

The above algorithm was designed and developed for simple obstacle avoidance, but works effectively in most cases. The robot is still prone to collisions. For a more robust obstacle avoidance solution, the algorithm needs to be enhanced by using much more data sampled through different tools such as computer image processing and laser technologies. A future version of this robot will contain these technologies and more to provide a truly autonomous solution.

Torso

The upper body torso is a collection of servos to provide joint mechanics connected by brackets. An Adafruit 16-Channel Servo Driver (Adafruit, 2014) was selected since the Arduino Mega does not have enough Pulse Width Modulation (PWM) ports available. The torso drive was implemented to have one command per servo. To emulate torso movements, many statements commanding servos' movements were written into functions such as *waveHand*. When the *waveHand* function is called, the servos are commanded to move to simulate its hand waving. The torso was added more for curiosity than an actual solution. In the future, manipulation of an arm, grabber or other tool can be useful for hazardous situations where it would be too risky to send a human or animal.

Figure 2. Completed Robot Platform

User Interface

The main purpose of the user interface (UI) is to allow the user to control the robot at the user's discretion. Its primary functions were to include streaming the video from the webcam, switching between autonomous and manual modes, and controlling the robot. By having a video stream from the camera, the user would be able to see what is viewable to the robot and can make adjustments in the event that there is an obstacle that is not easily avoided. Additionally this allows the user to control the robot without having to be too close in proximity.

Figure 3. Robot User Interface

The UI was designed and developed by using Python version 2.7 with PyQT to include a single screen to view the video stream as well as 6 buttons: forward, turn left, turn right, reverse, autonomous mode on/off, and play/stop for the video

stream. Currently the UI runs on a computer, but in the future, we intend to implement a mobile platform version. Figure 3 shows the current design of the UI.

To display the video stream, the UI utilized VLC media player. It was selected because it allows video content streaming through URL or IP addresses. Streaming the video through VLC player was made possible by using the Python VLC package. Currently, the UI is able to start and stop the video feed. To stream the video feed a user simply has to press the Play button, and to stop the video feed the user needs to press the Stop button. A WIFI connection is required in order for the UI to be able to stream the video. A side effect of this design is that there is a delay from when the stream begins. In the future, we intend to implement streaming capability with less of delay. It is also noteworthy, that the UI does not turn the camera on or off, but instead, it plays or stops viewing the stream.

Figure 4 shows the pseudo code of the robot control algorithm:

```
IF user presses play button THEN
        IF video stream is on THEN
                video stream is turned off
        ELSE
                video stream is turned on
ENDIF

IF user presses autonomous mode button THEN
        IF autonomous mode is on THEN
                autonomous mode is turned off
                WHILE autonomous mode is off THEN
                                CASE forward is pressed: robot moves forward
                                CASE reverse is pressed: robot moves in reverse
                                CASE turn left is pressed: robot turns left
                                CASE turn right is pressed: robot turns right
                                CASE autonomous mode button is pressed: autonomous mode is on
                                DEFAULT: do nothing
                                ENDCASE
                ENDWHILE
ENDIF
```

Figure 4. Robot Control Algorithm

Controlling the robot is intended to be achieved by using the UI to call movement-based methods when set to manual mode. Initially this concept was going to be achieved by using the webserver on the Raspberry Pi but there was too large of a marginal delay between pressing buttons and movement. In the future, we intend to fully implement controlling the robot with minimal latency.

CONCLUSION

In this paper, we presented a multipurpose educational robotics platform designed and implemented by combining various currently available robotics technologies. Its main functions include differential drive, autonomous obstacle avoidance, torso and arms movement, real-time video streaming, and remote control. This robotics platform was developed to engage students at middle school, high school, and college levels in science, engineering, and computer programming principles and in near future will be used in robot programming course in Intelligent Systems concentration at a college. In the course, students will be working on various projects for enhancing current functions and implementing new functions. The future work includes a series of experiments to examine whether the robot platform really serves for the intended purpose.

REFERENCES

1. Adafruit. (2014). *Adafruit 16-Channel Servo Driver with Arduino.* Retrieved from https://learn.adafruit.com/downloads/pdf/16-channel-pwm-servo-driver.pdf
2. Barker, B. S., & Ansorge, J. (2006). Using robotics as an educational tool in 4-H. *Journal of Extension, 44*(5).

3. Chattopadhyay, A., & Sellman, G. (2013). Developing the cellbot learning framework (CLF)-An interdisciplinary model for integrating mobile computing with robotics to innovate STEM education and outreach. *IEEE Frontiers in Education Conference*, (pp. 1290-1292).
4. Cielniak, G., Bellotto, N., & Duckett, T. (2013). Integrating mobile robotics and vision with undergraduate computer science. *IEEE Transactions on Education, 56*(1), 48-53.
5. Connaughton, R., & Modlin, M. (2009). A modular and extendable robotics platform for education. *Proceedings of 39th ASEE/IEEE Frontiers in Education Conference*, (pp. 1-4). San Antonio.
6. Dodds, Z., Greenwald, L., Howard, A., Tejada, S., & Weinberg, J. (2006). Components, curriculum, and community: Robots and robotics in undergraduate ai education. *AI magazine, 27*(1), 11.
7. Eckel, T. (2014). *NewPing Library for Arduino.* Retrieved from ARDUINO: http://playground.arduino.cc/Code/NewPing
8. Habib, M. A. (2012). Starting a Robotics Program in Your County. *Journal of Extension, 50*(2).
9. Kurkovsky, S. (2014). Interdisciplinary connections in a mobile computing and robotics course. *Proceedings of the 2014 conference on Innovation & technology in computer science education*, (pp. 309-314).
10. Nielsen, C. W., Goodrich, M. A., & Ricks, R. W. (2007). Ecological interfaces for improving mobile robot teleoperation. *IEEE Transactions on Robotics, 23*(5), 927-941.
11. Pololu. (2015). *Pololu - Qik 2s9v1 User's Guide.* Retrieved from http://www.pololu.com/docs/pdf/0J25/qik_2s9v1.pdf
12. Ruzzenente, M., Koo, M., Nielsen, K., Grespan, L., & Fiorini, P. (2012). A review of robotics kits for tertiary education. *Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*, (pp. 152-162).
13. Saad, A. (2004). Mobile robotics as the platform for undergraduate capstone electrical and computer engineering design projects. *IEEE Frontiers in Education, 3*, pp. S2G–7–11.
14. Yu, L., Tsui, P. W., Zhou, Q., & Hu, H. (2001). A web-based telerobotic system for research and education at Essex. *Proceedings of 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2001. , 1*, pp. 37-42.