Proceedings of the Fourteenth Annual Early Career Technical Conference
The University of Alabama, Birmingham ECTC 2014
November 1 – 2, 2014 - Birmingham, Alabama USA

LOW-COST PLATFORM FOR AUTONOMOUS GROUND VEHICLE RESEARCH

Nikhil Ollukaren, Dr. Kevin McFall
Southern Polytechnic State University
Marietta, Georgia, United States of America

ABSTRACT

This research paper investigates what it takes to design an affordable autonomous ground vehicle platform from the ground up. Using a handful amount of readily available parts, a go-kart sized vehicle was built and functional for approximately $400. Using an independent direct drive rear wheel system the vehicle can achieve a fixed axis rotation. Using vision algorithms loaded onto a Raspberry Pi the robot is able to detect a red target and send commands to the Arduino. The Arduino controls the motion logic and allows the vehicle to follow the target. The vehicle can also be driven manually using a hand held controller. This platform will allow for expansion of the project into more complex tasks.

INTRODUCTION

Recent years have seen an explosion in research into autonomous vehicles [1–7]. This project aims to design and build a platform that uses visual data to navigate its surroundings. Using simple and affordable microcontroller boards, a vehicle was created that was able to track and follow a specified moving target using visual algorithms to supply motion logic. Other research teams have successfully designed systems that can, using a stationary camera, track specified shapes and colors. And still others have mounted multisensory arrays onto automobiles and achieved controlled environment autonomous travel such as Google's driverless car [8]. As far as commercial success a few civilian vehicles have driver assistive features but no fully autonomous automobile is available to the consumer. So far a few states have legalized driverless vehicles on state roads and others have legislation in the works. Through this research, technologies can be developed that can utilize this new market space and radically change the transportation industry.

VEHICLE DESIGN

In order to test visual tracking code for the autonomous vehicle, a physical platform was required to mount the necessary mechanical and electrical hardware. Other requirements included a sturdy design, high maneuverability, proper clearance, and height specifications, which should be achieved with a build cost as low as possible. Additionally a control device would allow for manual driving of the vehicle and to toggle into autonomous mode or kill power to the driver motors entirely.

Looking into the collection of readily available parts allowed the initial design of the "go-kart" to take shape with old mini-bike tires as the center of the build. By recycling the tires the cost for the rear assembly would be minimized while retaining large sturdy rear wheels. Investing early in a pair of 2.5 inch CIM Brushed DC Motors and Talon SR motor controllers provided sufficient motor power and allow implementation of direct drive rear wheels which would run independent from one another. A single motor directly connected with a coupling to a wheel is captured in Figure 1.

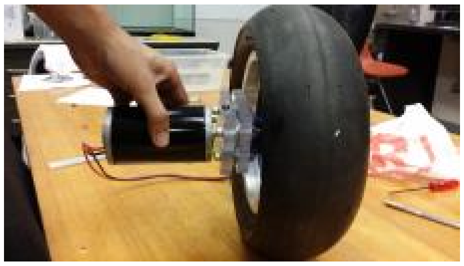
Figure 1. Motor-Wheel Assembly

By placing 3 inch castor wheels on the front of the vehicle, it would have the ability to rotate on a fixed axis of rotation centered at the back by reversing the direction of the motors as shown in Figure 2. This gives the vehicle a high degree of maneuverability in tight spaces while also giving the vehicle the necessary clearance to get over small objects in its path.

The frame of the go-kart was cut out of plywood and the castors were mounted on the front. The front corners of the go cart were rounded so that the cart would not get caught on edges. L-brackets were used to secure the motors to the board and couplings had to be made to mate an 8 mm shaft with 2 mm key to a three bolt pattern on the mini bike wheel.

The coupling system took considerably more time than was originally allotted due to design revisions. In hindsight, less emphasis should have been put on using the custom mini bike

(Do not add page numbers; editor will add sequential page numbers in the ECTC Journal.)

tires and a wheel that supported a key shaft should have been considered.

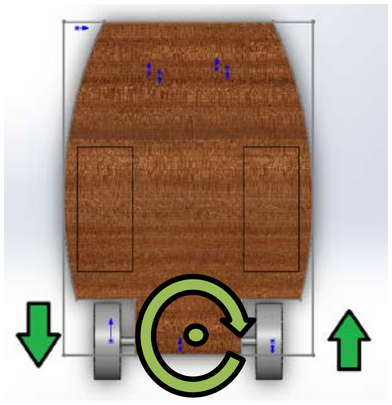
Figure 2. Axis of rotation

The final structure to be implemented was the stand for the Raspberry Pi camera box which was to be placed three feet above the ground to simulate eye level for a typical car driver. Using a PVC pipe and small L-brackets the stand was secured at the front of the vehicle, between the castors. Using a bike phone mount allows us to quickly attach and detach and adjust elevation of the camera as seen in Figure 3 below.

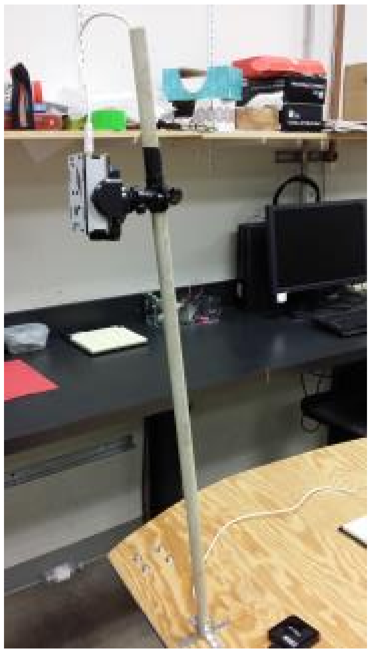
Figure 3. Stand with Raspberry Pi Mount

The vehicle controller was cut out of plastic using a Dremel tool and mounting holes and button spaces where drilled out and assembled. Potentiometers connected to joysticks were used to read in the throttle values for each motor. Three toggle switches were included for different modes such as the kill switch, autonomous mode and a free toggle for future functionality. The completed controller can be seen in Figure 4. The controller is tethered to the vehicle so the operator can continuously monitor the vehicle and activate the kill switch should the machine malfunction.

The controller also has room to add more switches and possibly an LED screen for feedback from the vehicle. One specific improvement for the next version of the controller would be a dead man handle switch which would act as a secondary kill switch should the vehicle pull the controller out of the operator’s hands.

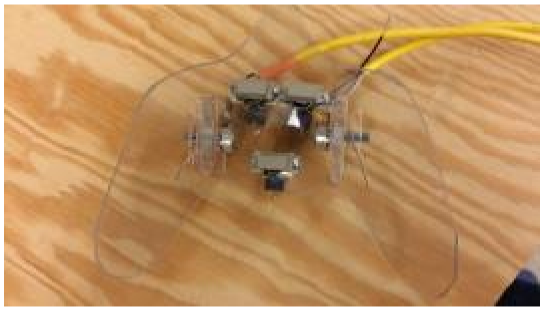
Figure 4. Vehicle Controller

The completed platform met all of the design specifications while allowing there to be room on the vehicle for expansion of the current project. By recycling older components into the design we were able to keep assembly costs under $400 for the entire build. Figure 5 shows the completed platform with all of the components wired.

MICROCONTROLLER PROGRAMMING

While versatile and simple to use, microcontrollers like the Arduino and applications processors like the Raspberry Pi are limited in their processing speed especially if parallel processing threads are required. For this reason, an Arduino Uno was chosen to send pulse width modulation (PWM) signals to the motor controllers and the Raspberry Pi for processing the camera image in order to compartmentalize the computing tasks. The Pi was chosen for its camera module that connects directly to the board with a dedicated camera port. Both the Pi and camera are protected by placing them in a commercially available enclosure.

The Pi collects visual information from the camera, processes it, and sends serial data commands to the Arduino via USB cable as illustrated in Figure 6. The role of the Arduino is to read inputs from the controller and Pi, and write appropriate PWM commands to the motor controllers as described in Figure 7. The PWM signals to the Talon motor controllers consist of 1 to 2 ms pulses periodically with a 20 ms period. Full counter-clockwise is achieved with a 1 ms pulse where 1.5

ms is neutral and 2 ms full clockwise rotation. The software limits PWM output between 1.1 and 1.8 ms in order to operate the tethered vehicle at reasonable speeds.

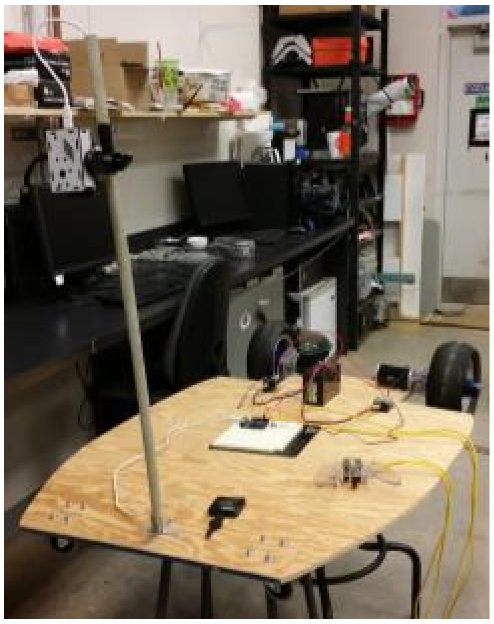

Figure 5. Completed Vehicle

Powering both the Arduino and Pi with a single battery was more complicated than anticipated. Most desirable would be to power the Arduino with a battery and subsequently the Pi via the USB serial connection; attaching a battery to the vehicle chassis along with the Arduino is simpler compared with adding one to the Pi mounted on a post. Constructing a supply for the Arduino from a 9 V battery, battery snap, and 2.1 mm power plug is straightforward, and takes advantage of the built-in voltage regulator on the Arduino. However, a diode in the Arduino circuitry [9] allows powering the Arduino via USB but prevents it from supplying power to USB connected devices. As a workaround, a USB cable was modified with its power wire connected to the 5 V output on the Arduino rather than to the USB connector. Power was supplied to the Pi, although none of the USB port would activate, not even serial communication across the cable powering it from the Arduino. The final design places a battery pack with the Pi instead, which powers the Arduino via a standard USB cable. A Bytech portable battery charger was used to deliver the requisite 5 V to the Pi, which has no internal voltage regulator and therefore should not be powered directly with batteries. The overall system schematic of how all the components are connected is presented in Figure 8. The source code for the Python script running on the Pi and the Arduino sketch appear in Appendix A and B, respectively.

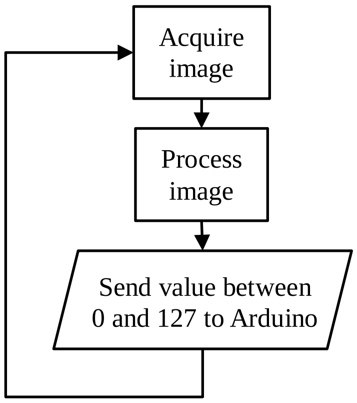

Figure 6. Flow chart for Raspberry Pi programming.

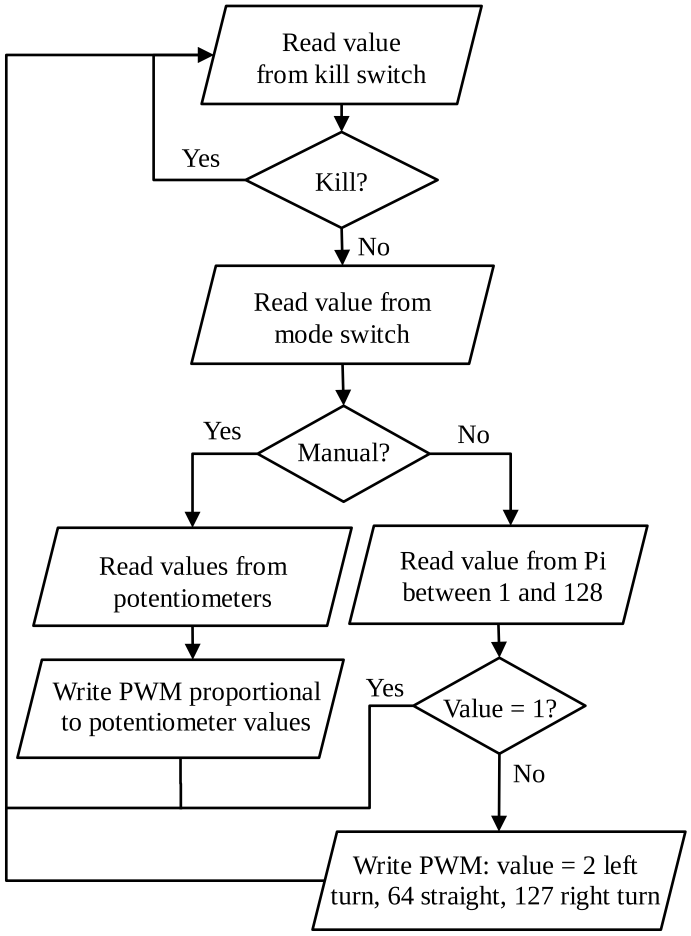
Read value from kill switch
Yes
Kill?
No
Read value from mode switch
Yes
Manual?
No
Read values from potentiometers
Read value from Pi between 1 and 128
Write PWM proportional to potentiometer values
Yes
Value = 1?
No
Write PWM: value = 2 left turn, 64 straight, 127 right turn

Figure 7. Flow chart for Arduino programming.

(Do not add page numbers; editor will add sequential page numbers in the ECTC Journal.)

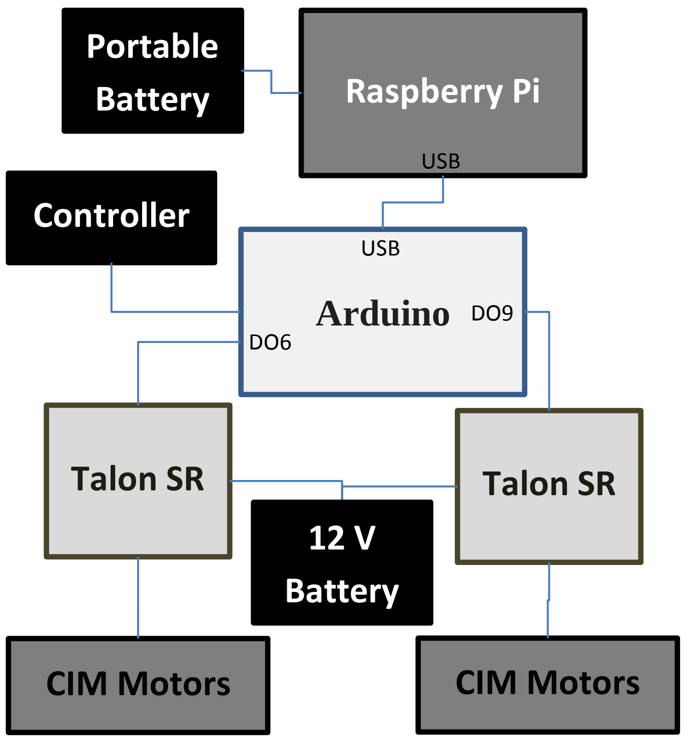

Figure 8. System schematic

IMAGE PROCESSING

The eventual goal of this project is to equip the autonomous vehicle with sophisticated image processing algorithms to offer obstacle avoidance and detection of road boundaries for steering control. As a proof of concept, the prototype is programmed to detect and follow a sheet of red paper. The picamera Python library [10] can acquire images at 30 frames per second (FPS), but the overhead of processing the image generally reduces the rate to 2-5 FPS [11], [12]. but the simple code tested using `capture_continuous` yielded only approximately 1 FPS, which is sufficient for a prototype but obviously must be improved upon for a relevant autonomous vehicle.

The OpenCV computer vision library [13], [14] is used to encode the raw image input into a useful matrix in order to create a binary image true for pixels with red components over 68% and with blue and green components under 47%. A non-zero command is sent via USB to the Arduino if the number of true pixels in the binary image exceeds a particular threshold. The value sent is proportional to the average horizontal position of the true pixels with 1 corresponding to the left side, 64 to middle, and 127 to the right side. Engaging the kill switch on the controller prevents the vehicle from following spurious red objects in the background.

The image acquisition and processing script is run automatically in the background (using the & tag) during startup by executing it in the /etc/rc.local file just before the exit command.

RESULTS AND CONCLUSION

Throughout this build a lot has been learned about the mechanical structure, electrical hardware and software that go into creating an autonomous platform. The limitations of the current design were more apparent as the semester wore on and will play into how the next iteration will proceed. However, the platform that was completed this semester meets all the design goals at an astonishingly low price considering how much a kit would have cost to achieve the same functionality. The added advantage of intimately knowing every inch of the vehicle will allow the expansion of the system for more tasks with ease.

More importantly the technology being investigated could be used in industry to change the way that transportation is done both commercially and individually. The code created in this project could be used to create unmanned vehicle caravans that would only need one driver to lead. As the project evolves the vision code will assist the vehicle in recognizing lanes and other marks on the road. This will allow for the vehicle to autonomously take over command at the operators request in approved situations. An example of such a situation is dropping off the occupant and finding a parking spot and self-parking and returning to the owner when called by car keys or mobile device. Another situation could be driving the high way portion on a road trip. This would allow the occupants of the vehicle to relax and enjoy the areas they are driving through. Undoubtedly the market and field of autonomous robots has begun to expand and change rapidly. With the changes in upcoming legislation and public demand for these vehicles research in this field is essential to created safe and reliable products.

(Do not add page numbers; editor will add sequential page numbers in the ECTC Journal.)

REFERENCES

[1] M. A. Zakaria, H. Zamzuri, R. Mamat, and S. A. Mazlan, “A Path Tracking Algorithm Using Future Prediction Control with Spike Detection for an Autonomous Vehicle Robot,” *International Journal of Advanced Robotic Systems*, vol. 10, pp. 1–9, Aug. 2013.

[2] R. Domínguez, J. Alonso, E. Onieva, and C. González, “A transferable belief model applied to LIDAR perception for autonomous vehicles,” *Integrated Computer-Aided Engineering*, vol. 20, no. 3, pp. 289–302, Sep. 2013.

[3] Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, and Tao Mei, “Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID,” *International Journal of Advanced Robotic Systems*, vol. 9, pp. 1–11, Jul. 2012.

[4] R. Kala and K. Warwick, “Motion planning of autonomous vehicles in a non-autonomous vehicle environment without speed lanes,” *ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE*, vol. 26, no. 5–6, pp. 1588–1601, Jun. 2013.

[5] M. C. Best, “Optimisation of high-speed crash avoidance in autonomous vehicles,” *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 4, pp. 337–354, Sep. 2012.

[6] S. Raju, K. Sanjay, T. Sathish, and B. Madhini, “Semi Autonomous Vehicle To Prevent Accident,” *International Journal of Technology and Emerging Engineering Research*, vol. 2, no. 5, 2014.

[7] A. Broggi, P. Cerri, M. Felisa, M. C. Laghi, L. Mazzei, and P. P. Porta, “The VisLab Intercontinental Autonomous Challenge: an extensive test for a platoon of intelligent vehicles,” *International Journal of Vehicle Autonomous Systems*, vol. 10, no. 3, pp. 147–164, Jun. 2012.

[8] G. Erico, “How Google’s Self-Driving Car Works,” *IEEE Spectrum*, vol. 18, 2013.

[9] “Arduino Uno Rev 3 Schematic.” [Online]. Available: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf. [Accessed: 05-Jul-2014].

[10] “Documentation for the picamera.” [Online]. Available: http://picamera.readthedocs.org/en/release-1.0/. [Accessed: 06-Jul-2014].

[11] C. Venables, “Multirotor Unmanned Aerial Vehicle Autonomous Operation in an Industrial Environment using On-board Image Processing,” Capstone thesis project, University of Western Australia, 2013.

[12] I. Petrov, “Raspberry Pi based System for Visual Detection of Fluid Level,” Capstone thesis project, Tallinn University of Technology, 2014.

[13] K. Pulli, A. Baksheev, K. Kornyakov, and V. Erumihov, “Real-Time Computer Vision with OpenCV,” *Communications of the ACM*, vol. 55, no. 6, pp. 61–69, Jun. 2012.

[14] S. Brahmbhatt, “Embedded Computer Vision: Running OpenCV Programs on the Raspberry Pi,” in *Practical OpenCV*, Apress, 2013, pp. 201–218.

APPENDIX A – RASPBERRY PI PYTHON SCRIPT

```
import io
import cv2
import numpy as np
from serial import Serial
rowRes = 60
colRes = 80
ser = Serial('/dev/ttyACM0')
 with picamera.PiCamera() as camera:
   camera.resolution = (colRes, rowRes)
   camera.framerate = 30
   stream = io.BytesIO()
   for foo in camera.capture_continuous
                (stream, format='jpeg'):
    stream.truncate()
    stream.seek(0)
    data = np.fromstring
      (stream.getvalue(), dtype=np.uint8)
    im = cv2.imdecode(data,1)
    B = im[:,:,0] // Blue
    G = im[:,:,1] // Green
    R = im[:,:,2] // Red
    ind = (R>175) & (B<120) & (G<120)
    x = 0
    count = 0
    for row in range(0,rowRes):
     for col in range(0,colRes)
      if(ind[row,col] == True):
       x += col
       count += 1
    x = x/(count+1) // Avoid divide 0
    cover=float(count)/rowRes/colRes*100
    scale = int(float(x+1)/colrRes*128)
    if(cover > 0.2):
     ser.write(str(unichr(scale
    else:
     ser.write(str(unichr(1)))
```

APPENDIX B – ARDUINO SKETCH

```
#define baseSpeed 200
#define maxSpeed  400
int val = 1;
int newVal, time, buffer;
int right, left;
void setup() {
  pinMode(6, OUTPUT); // Left motor
  pinMode(9, OUTPUT); // Right motor
  pinMode(5, INPUT);  // Mode switch
  pinMode(4, INPUT);  // Kill switch
  Serial.begin(9600);
}
void loop() {
 if(digitalRead(4) == LOW) {
  if(digitalRead(5) == LOW) {
```

(Do not add page numbers; editor will add sequential page numbers in the ECTC Journal.)

```
        newVal = 0;
        while((buffer=Serial.read())>=0)
          newVal = buffer;
        if(newVal > 0)
          val = newVal;
        if(val == 1) {
          digitalWrite(6,HIGH);
          delayMicroseconds(1500);
          digitalWrite(6,LOW);
          delayMicroseconds(20000-1500);
        } else {
          if(val < 64) { // Move left
            time = map(val,64,0,
                        baseSpeed,maxSpeed);
            digitalWrite(6,HIGH);
            digitalWrite(9,HIGH);
            delayMicroseconds(1500-
                                    baseSpeed);
            digitalWrite(6,LOW);
            delayMicroseconds(baseSpeed +
                                       time);
            digitalWrite(9,LOW);
            delayMicroseconds(20000 –
                                 1500 - time);
          } else { // Move right
            time = map(val, 64, 128, 0,
                        maxSpeed-baseSpeed);
            digitalWrite(6,HIGH);
            digitalWrite(9,HIGH
            delayMicroseconds(1500-time);
            digitalWrite(6,LOW);
            delayMicroseconds(time +
                                    baseSpeed);
            digitalWrite(9,LOW);
            delayMicroseconds(20000 –
                                1500 - time);
          }
        }
      } else { // Manual mode
        left  = map(analogRead(1),0,1023,
                        1500,1500+maxSpeed);
        right = map(analogRead(2),0,1023,
                        1500,1500+maxSpeed);
        digitalWrite(6,HIGH);
        digitalWrite(9,HIGH);
        delayMicroseconds(1500 - left);
        digitalWrite(6,LOW);
        delayMicroseconds(left + right);
        digitalWrite(9,LOW);
        delayMicroseconds(20000 - 1500 –
                                  right);
      }
    }
  }
```