Project Proposal:
Collision Avoidance with Use of Image Processing with Arduino-to-Android Platform Interaction

By Charles Norona and Marcorel Atilus
COT 5930 – Android Projects
Professor Ravi Shankar
Fall 2010

Abstract

The Collision Avoidance project is meant to integrate the means for allowing robots, or rovers, an ability to avoid obstacles with the use of vision systems. In this case a small Arduino-based rover will have a hardware camera whose images will be sent via Bluetooth to an Android mobile device for image processing who will then send back information to the rover so that it can determine how to behave. The implementation will consist of most of the stages of the machine vision system (i.e. acquisition, pre-processing, segmentation, feature extraction, and classification) as well as some behavioral logic that the rover will need to execute. With the completion of this project a blueprint for Android-to-Arduino interaction and collision avoidance with the use of image processing will be established.

Background

The field of robotics combines many technological disciplines and philosophies. Not only are there electrical and mechanical considerations but also computer science mechanisms that play just as significant a role. The Collision Avoidance implementation will take advantage of embedded system implementations based on the Arduino microcontroller, ATmega328P, with interactivity to an Android-platform application which will be responsible for assisting the rover in avoiding objects in its path of motion. Previous implementations that focus on subsets of the total functionality of this project have been investigated.

The robotic collision avoidance implementation will be inspired by the works of an article called "Neutral network Robot using Microcontroller Atmega32". The completion of this project required capacity and application on both hardware and software. To construct this robot, we first need to build the custom prototype board by using a ping sensor as neutral input, implementing two DC motors for the motion of the robot, and batteries to provide power supply to the Microcontroller and the rest of the electrical devices.

For their master's thesis efforts, Sebastian Olsson and Philip Åkesson made significant efforts in implementing image processing techniques such as Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) on the Android platform. SIFT and SURF are both used for detecting and classifying image objects which Olsson and Åkesson proved to be possible on the Android platform as well as in a distributed configuration involving PCs. In addition to this, the inspiration to have the image processing stages distributed to the mobile phone is based on their using PCs to distribute the workload of image processing tasks acquired by the mobile device [Olsson and Åkesson].

The Collision Avoidance implementation will also take advantage of Bluetooth communications in order to transmit image information from the Arduino robot to an Android handset. Previous implementations have been made such as that described in Joseph Gundel's and Brian Chamba's "Blue-tooth Setup" article at robotics.fau.edu [Gundel].

Methods

The implementation of this project involves Android application programming and Arduino hardware and software implementation. The Android half of the design comprises of Bluetooth programming, digital image processing, and some basic input controls. In the second half of the implementation, on the Arduino platform, is hardware components such as physical, mechanical components to allow the Arduino rover to move as well as an Arduino microcontroller with a camera interface and ping sensors for inputs. The software on the microcontroller is largely responsible for movement control and collision avoidance logic with some Bluetooth communication sequences.

The person responsible for the Android application is Charles Norona while Marcorel Atilus will ensure that the hardware and most of the software for the hardware platform is completed.

References

Olsson, S. and Åkesson, P. "Distributed Mobile Computer Vision and Applications on the Android Platform." Master's Thesis Paper. Lund University. 2009.

"Neutral network Robot using Microcontroller Atmega32" http://www.circuitlake.com/neural-network-robot-using-microcontroller-atmega-32.html

Gundel, J. "The Blue-tooth Setup." http://robotics.fau.edu/2010/10/10/76/. Oct. 14th, 2010.

Equipment

- DMM
- Screw drivers
- Arduino Program
- Oscilloscope
- Soldering Iron
- Laptop
- "Extra hand holder"
- USB cable
- Tin Solder
- Pspice
- Cell phone
- Hot glue

Components and Materials

- Micro-controller
- 2 DC Motors
- 1 ping Sensor
- 1 voltage regulator
- Wire
- H-bridge m. driver
- EEPROM
- LEDs
- Transistor
- Cmos Camera
- Battery Holder
- Batteries

Approach - Initial ideas

Sensor

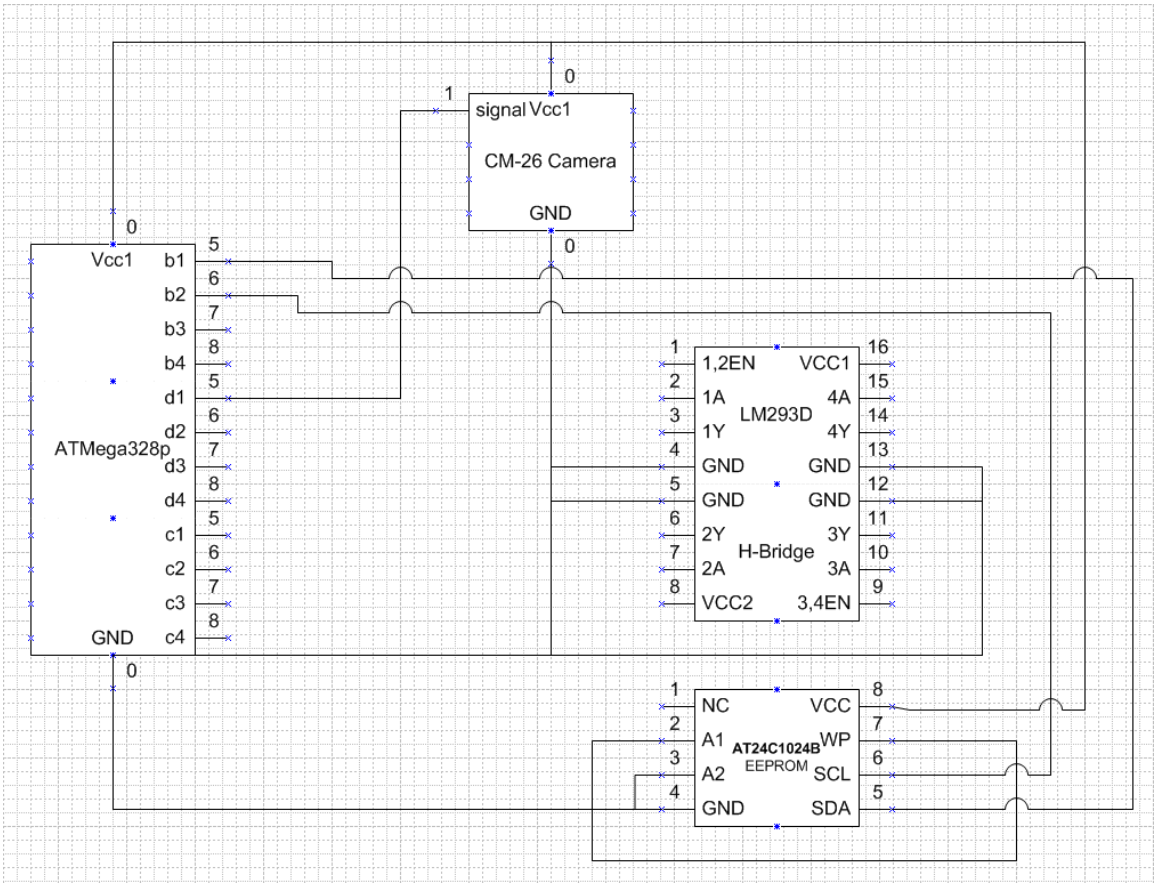
Microcontroller

Robot

Collision avoidance

Challenges

Wiring – Soldering Circuits Together and programming



Programming Code

```
//Test Program written for Arduino Robot v1.0
//Marcorel Atilus, Charles Norona

const int pingPin = 7;       // ping trigger pin
const int motor1aPin = 3;    // H-bridge motor+ 1 (pin 2, 1A)
const int motor1bPin = 4;    // H-bridge motor- 1 (pin 7, 2A)
const int motor2aPin = 5;    // H-bridge motor+ 2 (pin 15, 1A)
const int motor2bPin = 6;    // H-bridge motor- 2 (pin 10, 2A)
const int enablePin = 9;    // H-bridge enable pin


void setup() {

  // set all the other pins as outputs:
  Serial.begin(9600);
 pinMode(motor1aPin, OUTPUT);
  pinMode(motor1bPin, OUTPUT);
  pinMode(motor2aPin, OUTPUT);
  pinMode(motor2bPin, OUTPUT);
  pinMode(enablePin, OUTPUT);


}

void loop() {

  long duration, inches;

  duration = readSensor();

  inches = convertDurationToInches(duration);

  Serial.println(inches);

  controlMotor(inches);
}

long convertDurationToInches(long duration)
{
  return duration / 147;
}

long readSensor()
{
```

```
 pinMode(pingPin, OUTPUT);
digitalWrite(pingPin, LOW);
delayMicroseconds(2);
digitalWrite(pingPin, HIGH);
delayMicroseconds(15);
digitalWrite(pingPin, LOW);

// The same pin is used to read the signal from the PING))): a HIGH
// pulse whose duration is the time (in microseconds) from the sending
// of the ping to the reception of its echo off of an object.
pinMode(pingPin, INPUT);
return pulseIn(pingPin, HIGH);
}

void controlMotor(int inches)
{
   digitalWrite(motor1aPin, HIGH);
   digitalWrite(motor1bPin, LOW);
   digitalWrite(motor2aPin, HIGH);
   digitalWrite(motor2bPin, LOW);
 analogWrite(enablePin, 0);
 if(inches > 10)
 {
  analogWrite(enablePin,250); //enable motors 60%duty cycle
 }
 else
 {
  //if the space between robots is less than 10 inches
  // this robot will rotate until it finds an ampty space
  analogWrite(enablePin, 200);
  digitalWrite(motor1aPin, HIGH);
   digitalWrite(motor1bPin, LOW);
  digitalWrite(motor2aPin, LOW);
   digitalWrite(motor2bPin, HIGH);
 }
}
```