C++ PROGRAM FOR DRIVING OF AN AGRICOL ROBOT

STELIAN-VALENTIN CASAVELA[1]

Abstract: This robot is projected to participate to the international FieldRobotEvent, which will take place in Venlo (The Netherlands) in 2013. This event was founded by Wageningen University in 2003. The use of simple hardware (infrared or ultrasonic sensors) instead of expensive and sophisticated equipment (PCs, laser, GPS), will get more points. In maximum three minutes, the robot must travel between long curved rows of potted flowers. On the end of a row, the equipment must return on the next row. First, as an imposed criterion by the officials, it will be told whether this starting is on the left side of the land or on the right side. Some potted flowers are taken out of the land. So, the rows may contain gaps or dead ends. The robot will have to bypass them or to turn around. This is another challenge. The robot has six sensors, disposed on a circle, at sixty degrees one another and it is driven by a plc of type Arduino, which we programmed it in C++.

Key-words: Plc Arduino, controlled remote cars, numeric command, receiver, transmitter

1. THE REMOTE CONTROL (RC) CAR

1.1. Receiver-transmitter system

We put together an RC car and a radio command, using all these components to build a numerical command for an electric car. After this, we converted this into a robot, interfacing all equipment with a plc of type Arduino. We used a reduced-scale car for to investigate the vehicle dynamics. The mini car was bought from the company "New Era Models of Nashua, with the engine and drive train, suspension, wheels, servo motor systems, and fuel system already installed. The car was projected to resist for five hours of driving on combined field, powered by two battery. The suspension has been tuned to match with the electrics distribution. The dynamic stability control system has been adapted to the load of the wheels. The air conditioning's compressor starts to operate only if it is necessary and the brake system consists by an electric under pressure pump. Regenerative braking, where the dc engine acts as generator, it

[1] *Dr. Eng., lecturer at the University of Petrosani, cainegaston@yahoo.com*

means that is while the accelerator pedal is released. The heaviest part, the battery, is placed in the center of the chassis of the car, the dc motors being two in the front and two in the rear. Steering servo is also situated in the center, for optimum balance of weight and also for a great precision of the control and handling. The suspension with the double and flexible wishbone for the 4 wheels, which are independent, has a friction damper and is capable of all manner of driving conditions. The chassis of remote controlled car is one of the important components of its operation and reliability. The chassis is a platform, over it being placed the rest of the internal parts of the remote controlled car. Between these components are the motor and the radio receiver. The kind of chassis differs from model to model, but there are some important rules. The mini controlled remote cars, which contain an electric motor to work normal, are compound of plastic or carbon-fiber, for reducing the weight and increasing the performance. The hobby-fuel (a mixture of nitro, methane, and lubricants) used for mini cars controlled by radio, impose aluminum or metal chassis. This is because this type of motor with hobby-fuel, has a great weight and height. These metals can be added to the models' weight, but can also help with stability.

Fig 1. The remote controlled car

The receiver circuit is placed on the remote controlled car and is shown in figure 2. It contains an IR receiver module TSOP1738, two transistors of type BC548 (T6 and T7), resistors and capacitors. The dedicated integrated circuit, of type TSOP1738, receives light waves, in form of pulses modulated in width and converts them in electric signals, which being amplified, are needed for supplying of dc current to every dc motor. The simplest transmitter circuit contains two stable oscillators. See figure 3. The transistors T1 and T2 are used for to build the first oscillator, which generate a frequency of about 1.2 kHz. The second, with central components the transistors T3 and T4, produces about 38 kHz. IR LED1 is used to transmit the 38 kHz frequency. The pulses width modulation is obtained by mixing these two frequencies [1]. By mixing the two signals, given by stable oscillators, we obtained the Pulse Position Modulation (PPM), a method used for changing the distance between the pulses. Those pulses are integrated in the receiver circuit and, as a consequence, the motor is moving faster or slower. This is the simplest way to change the signal. The distance between the pulses is changed:

```
____l""""""""""l___l""""""""""l___l""""""""""l___l""""""""""l___ "Do nothing"
____l""""""""""l_l""""""""""l_l""""""""""l_l""""""""""l___ "Go faster"
____l""""""""""l____l""""""""""l____l""""""""""l____l""""""""""l___ "Go slower" [2]
```

There is a pair of receiver transmitter for every motor.

1.2. The numeric command

That is located on the back side of the remote controlled car, as it is seen in the figure 1. The receiver includes more channels like this in the figure 2 bellow, but more complex.

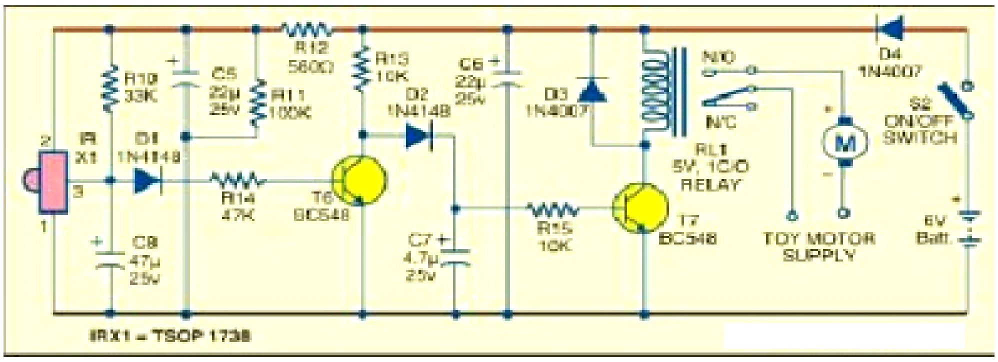

Fig. 2. The receiver circuit

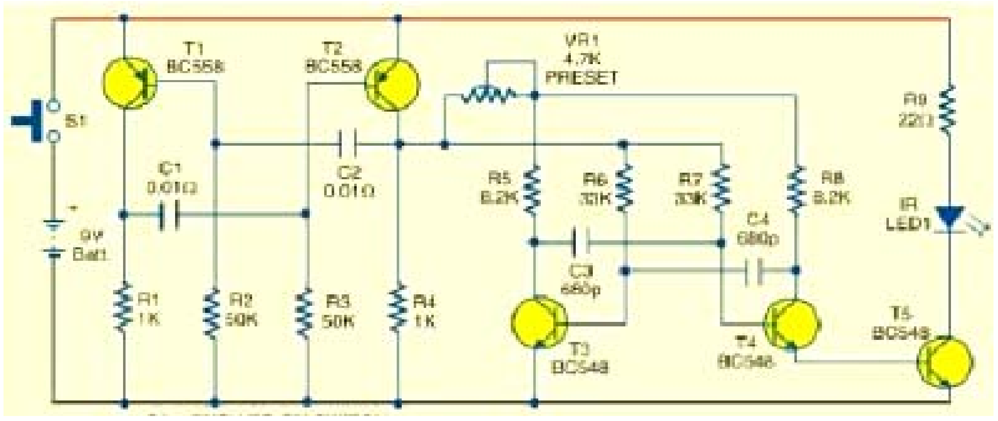

Fig. 3. The transmitter circuit

The left servo motor of the car must be connected to pin P13 of the component called numeric command, and its right servo motor must be connected to pin P12, as it is seen in figure 4 [3]. The transmitter should also send more complex signals, of some special keys, as it is shown in figure 5 [3].

2. CONVERING A REMOTE CONTROL (RC) CAR TO ROBOT

2.1. The ultrasonic sensors

Using ultrasonic sensors Ping Paralax type is to identify the position of environmental barriers in order to establish optimal strategies for avoiding them.

Another role is to measure the distance from some fixed points in the environment, to anchor its position in the workspace.

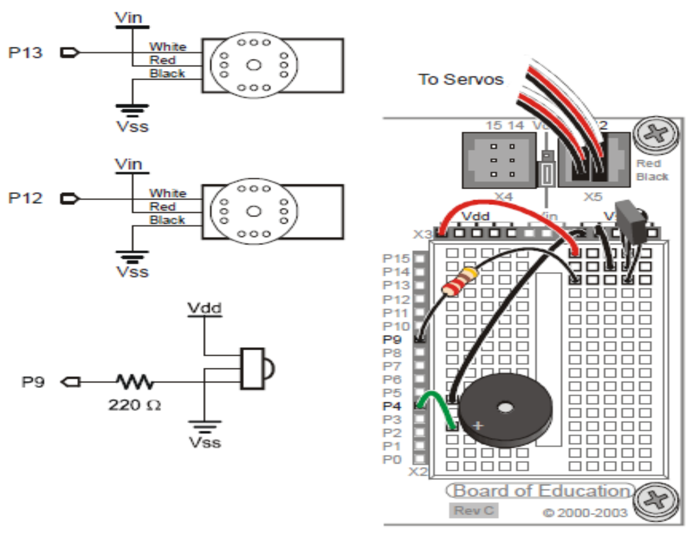

Fig. 4. The interface between numeric command and servo motors [3]

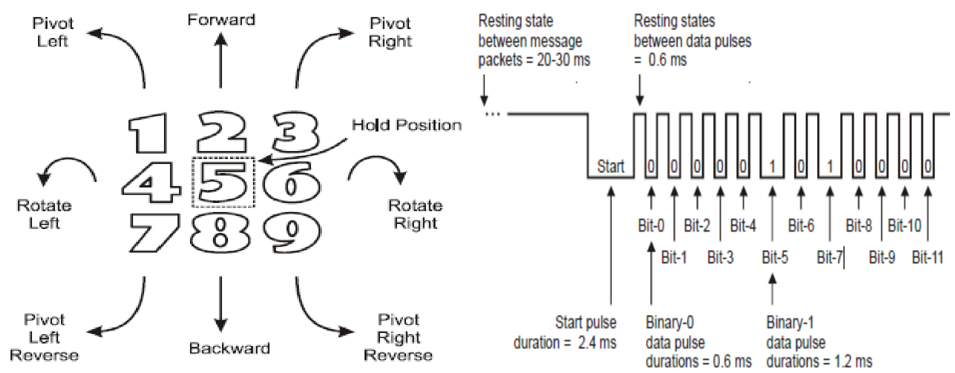
Pivot Left
Forward
Pivot Right
Hold Position
Rotate Left
Rotate Right
Pivot Left Reverse
Backward
Pivot Right Reverse
1 2 3 4 5 6 7 8 9 0
Resting state between message packets = 20-30 ms
Resting states between data pulses = 0.6 ms
Start
0 0 0 0 0 1 0 1 0 0 0 0
Bit-0 Bit-2 Bit-4 Bit-6 Bit-8 Bit-10
Bit-1 Bit-3 Bit-5 Bit-7 Bit-9 Bit-11
Start pulse duration = 2.4 ms
Binary-0 data pulse durations = 0.6 ms
Binary-1 data pulse durations = 1.2 ms

Fig. 5. All commands sent from the remote control transmitter and the SONY serial protocol for to transmit a code key

Specifications Range: 2 cm to 3 meters Voltage: 5V + / -10% (Absolute: Minimum 4.5V, 6V Maximum); Current: 30 mA, 35 mA maximum Interface: 3-wire (Vcc, mass signal). 20 mA power consumption .Input Trigger: TTL positive pulse, minimum 2 µs, 5 µs typical. Echo Pulse: Positive TTL pulse, 115 µs - 18.5 ms. Frequency: 40 kHz to 200 µs Dimensions: 22 mm *H* x 46 mm *W* x 16 mm *D*. [4].

A major advantage of this sensor is in that it only requires a single line I / O of a plc, as in figure 6.

The sensor is an ultrasonic range finder of type Parallax. It measures the distances from 2 cm up to 3m, of the nearest obstacle situated in front.

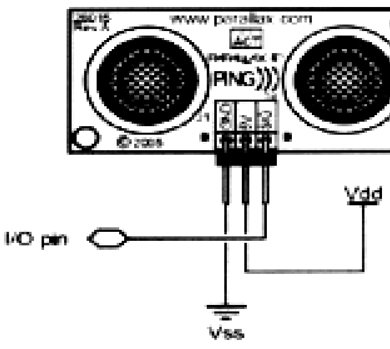

Fig. 6. Connections for sensor [4]

This sensor sends out a burst of ultrasound and listens the echo, after it hits an object. The plc with Arduino issues a short pulse to trigger the detection, then waits for a pulse on the same pin using the pulse In () function. The second pulse lasts the duration of the travel made by the ultrasound to arrive to the object and back to the sensor. Knowing the speed of sound, this time may be converted to distance. We mounted 6 sensors on a circle and we wrote a sequence of program, in C++ dedicated for Arduino, for measuring a number of six distances.

```
void distance_measurement(const int *pingPin, long *duration, long *cm)
//the distance cm result in centimeters...or inches:
{
for (int i = 0; i < 6; i = i + 1)
{
 // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
 // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
 pinMode(pingPin[i],OUTPUT);
 digitalWrite(pingPin[i],LOW);
 delayMicroseconds(2);
 digitalWrite(pingPin[i],HIGH);
 delayMicroseconds(5);
 digitalWrite(pingPin[i],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[i], INPUT);
 duration[i] = pulseIn(pingPin[i], HIGH);

 // convert the time into a distance
```

```
  // inches = microsecondsToInches(duration);
  cm[i] = microsecondsToCentimeters(duration[i]);

  // Serial.print(inches);
     Serial.print(cm[i]);
     Serial.print("  in  ");
     Serial.print("cm");
     Serial.println();

}//end for.
}
```

2.2. The interface between the plc Arduino and the remote control car

Fig7. The radio command

The radio command (figure 7) was mounted on the car and, over that, we mounted the circle with sensors and the kit type Arduino.

For setting up the interface, by radio command, we handled the two levers for moving the car in one of the six directions, each corresponding of a sensor position. For a small movement, we read four values: x1, y1 for first lever and x2, y2 for the second lever. These values are inputted in the sequence of the beginning of the program, called setup(), via the serial communication of type “com”, from the PC. That is the uploading step. See the program sequence.

```
void setup() {
  // initialize serial communication:
  Serial.begin(9600);
  pinMode(motor1plusPin, OUTPUT);// sets the pin 9 as //output
  pinMode(motor1minusPin, OUTPUT);   // sets the pin 10 //as output
  pinMode(motor2plusPin, OUTPUT);   // sets the pin 11 as //output
  pinMode(motor2minusPin, OUTPUT);   // sets the pin 12 //as output
  // send data only when you receive data:
      for (int i = 0; i < 6; i = i + 1)
    {
      if (Serial.available() > 0) {
            // read the incoming byte:
         sensor[i]= Serial.read();
         say what you got:
               Serial.print("I received i (integer) : ");
               Serial.print(sensor[i]);
           Serial.println();
      }//end if
         if (Serial.available() > 0) {
           // read the incoming byte:
         x1[i]= Serial.read();
         // say what you got:
         Serial.print("I received x1[i] : ");
           Serial.println(x1[i]);
           Serial.println();
      } //end if
              if (Serial.available() > 0) {
           // read the incoming byte:
      y1[i]= Serial.read();
         // say what you got:
         Serial.print("I received y1[i] : ");
           Serial.println(y1[i]);
           Serial.println();
         } //end if
         if (Serial.available() > 0) {
           // read the incoming byte:
         x2[i]= Serial.read();
         // say what you got:
         Serial.print("I received x2[i] : ");
           Serial.println(x2[i]);
            Serial.println();
      } //end if
       if (Serial.available() > 0) {
           // read the incoming byte:
         y2[i]= Serial.read();
         // say what you got:

         Serial.print("I received y2[i] : ");
           Serial.println(y2[i]);
           Serial.println();
      } //end if

}//end for
delay(100);
              }//end setup
```

2.3. The main program

That contains some specific functions: minimum (dir, cm), which calculates the minimum distance, that is meaning the distance to the nearest obstacle. It gives also the direction, after the sensor position. Reorder (dir, sensor, cm, x1, y1, x2, y2) reorders the arrays. Every array contains all these components of a movement. Always we reordered the sensors so that the direction of the movement is on the sensor called “zero”.

```
void reorder(int dir, int * sensor, long * cm, float * x1, float * y1,
float * x2, float * y2)//reorder arrays
{
int j=dir;
if (dir!=0){
for (int i = 0; i < 6-dir; i = i + 1)
{ init_s[i]=sensor[i];sensor[i]=sensor[j];
init_cm[i]=cm[i];cm[i]=cm[j]; init_x1[i]=x1[i]; x1[i]=x1[j];
init_y1[i]=y1[i];y1[i]=y1[j];
init_x2[i]=x2[i];x2[i]=x2[j];
init_y2[i]=y2[i]; y2[i]=y2[j];  j++;}
j=6;
for (int i = 0; i< dir; i = i + 1)
{ int k=j-dir; sensor[k]=init_s[i]; cm[k]=init_cm[i];
x1[k]=init_x1[i]; y1[k]=init_y1[i]; x2[k]=init_x2[i];
y2[k]=init_y2[i]; j++;
}
}
dir=0;
}
```

The car moving is effectively made by a function called mouveto.

```
void mouveto(int dir, float m, long *cm, float *x1, float  *y1,
float  *x2, float  *y2 , int motor1plusPin,
int motor1minusPin, int motor2plusPin,
int motor2minusPin )//mouve to sens direction
{
x1[dir]=m*x1[dir]; y1[dir]=m*y1[dir];
x2[dir]=m*x2[dir]; y2[dir]=m*y2[dir];

while (( (dir ==0) && (cm[0] <pow(10,(dir+1)))) || ((dir==1) && (cm[0]<100)))
// mouve to direction dir=0 or dir=1.
{
analogWrite(motor1plusPin, x1[dir]);// analogRead values go from 0 to 1023,
//..analogWrite values from 0 to 255.
analogWrite(motor1minusPin, y1[dir]);//similar
analogWrite(motor2plusPin, x2[dir]); //similar
analogWrite(motor2minusPin, y2[dir]); //similar
                    // wait for the robot to get there
}
}
```

The moving among obstacles is in a zig zag way, using the so called function or the method the first_minim.

```
void first_minim(int dir, long *cm)
{
int i=1;
while (i< 5)
{
if(cm[i] <= cm[i+1]){ i++;}
else
{
 dir = i+1;
}
}
}
```

The C++ program is structured in a modular manner and represents one of the author contributions in this project.

3. CONCLUSIONS

This type of robot may be extended in the domain of the remote control of the macro trucks or for tractors, which exists from some time, but driven by GPS. Our solution is cheaper and perfectible.

REFERENCES:

[1]. *,** http://www.electronicsforu.com/EFYLinux/circuit/June2011/CI-04_June11(1).pdf2
[2]. *,** http://wiki.openpilot.org/display/Doc/RC+Transmitter,+Receiver+and+Servos
[3]. *,** http://www.parallax.com/Portals/0/Downloads/docs/prod/sic/WebIR-%20v1.1.pdf
[4]. *,** http://www.scribd.com/doc/55081204/9/Senzori-ultrasonici
[5]. ***, http://www.arduino.cc/en/Tutorial/Ping