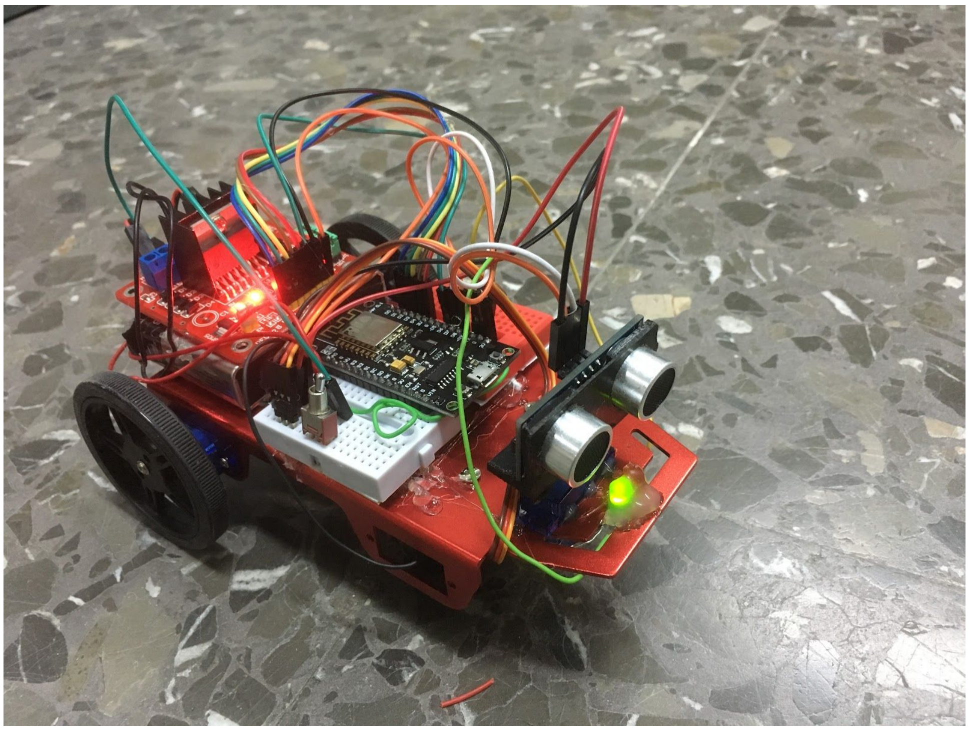
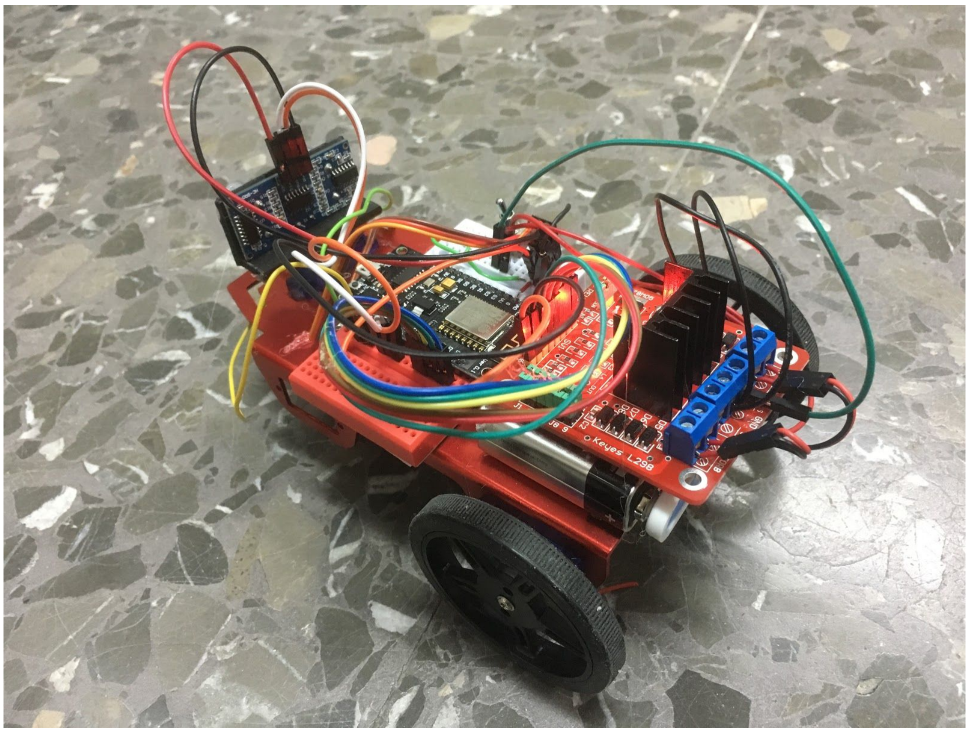
CLAUDEL Nathan
CHATAING Gauthier

UPM 115 : Physical computing

PROJECT REPORT - ShyCar





CLAUDEL Nathan
CHATAING Gauthier

Source code here

Synopsis :

Cars are getting smarter and smarter, and take more and more control away from the driver. Some cars are so smart that they even drive themselves. But what about RC cars ? Can we make a RC car that is a little bit smarter ? Can we reduce the control over an item whose only purpose is to be controlled ?

We can.

This is why we wanted to create a smart RC car, able to avoid obstacles by itself.

Final production :

We have created a car controlled by wifi. Thanks to a UDP connection, the operator of the car can control it with his computer and a processing program. The car is powered by two on-board 9V batteries, allowing it to attain a top speed of 500 meters per hour ! The car is turned on by flipping the switch.

At the front of the car, a ultrasound detector calculates the distance of the nearest obstacle. If the distance is higher than 30 centimeters, the car follows the order of the operator (Green led on). Otherwise, the car refuses to continue and will avoid the obstacle (Red led on). Similarly, when the operator wants to turn, the car will turn the detector to the direction of the turn to see if there is any obstacle on the way. If so, it will turn to the opposite direction. The closer the obstacle, the more violently it will react !

Propulsion :

The car is moved by one motor at each of its back wheels. Turns are done by differential power. A REV02 servo controls it, and our code to control the servo is the following:

```
void initMovement()
{
  … //initializes the pins
}

// Speed goes between -100 and 100
void setRightSpeed(signed char speed)
{
  … //rotates the right wheel
}

void setLeftSpeed(signed char speed)
{
  … //rotates the left wheel
}

void setMovement(signed char speed, signed char direction)
{
  //turn left
  if(direction < 0)
  {
    setRightSpeed(speed);
    setLeftSpeed((speed * (100 + direction))/100);
  }
  else
  {
    setLeftSpeed(speed);
    setRightSpeed((speed * (100 - direction))/100);
  }

  currentDirection = direction;
}
```

Arduino :

We used the nodeMCU board in order to get a wifi connection. This boards also has the benefit of having many GPIOs, which we took advantage of. In the end, we used all GPIOs the board has to offer, we even used the RX and TX pins to power the leds, disabling the serial connection.

Obstacle detection :

Our car detects obstacles with an ultrasounds sonar. The sonar sends ultrasounds impulsion than measure the delay of the echo. By multiplying the delay by the speed of sound and dividing the result by 2, we obtain the distance from the nearest obstacle. Before any turn, the servo pivots to put the sonar in the proper direction.

WIFI Communication :

To control the car remotely, we used the built-in wifi module of the nodeMCU. We used the UDP protocol because it is adapted to the real-time control we were looking for.

We created our protocol to communicate with the car: we send two chars in every packet. One is used to define the speed of the car (from -100 to 100), the other one gives the direction (from -100 for the leftmost turn to 100 for the rightmost turn).

Luckily, the wifi drivers for nodeMCU provides a UDP library which allowed un to easily communicate.

CLAUDEL Nathan
CHATAING Gauthier

Main structure:

The car fetches for a TCP packet which gives it the speed and rotation. Then it turns the detector in the proper direction and looks for an obstacle. If any obstacle is detected, the movement orders are adapted and the car stops obeying for a moment.

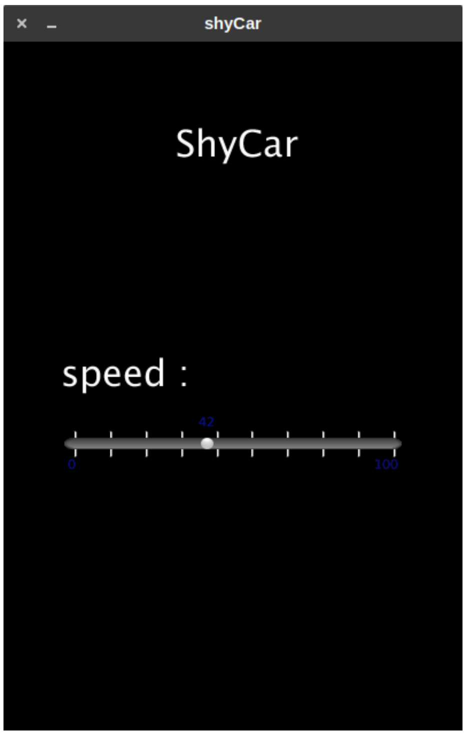
```
 void loop()
{
  // Receiving the packet
  signed char *code = receivePacket();
  commandSpeed = code[0];
  commandDirection = code[1];

  // Detector moving towards the direction of movement
  turnDetector(commandDirection);

  delay(10);

  // Obstacle detection
  int distance = getDistance();
  handleDistance(distance);

  // Updating movement
  setMovement(commandSpeed, commandDirection);
}
```

Remote control:

The remote we developed is a processing program.

The UI is very simple. You can control the car using the keyboard, and a slider allows you to adjust the speed of the car.

We could expand on this program, by for instance adding analog inputs, without changing the car's firmware.



Aborted features:

At first, we thought of using a RF connection to control the car with another arduino as a remote controller. However we chose to use wifi and a computer because the RF transmitter would have interfered with the servo controlling the proximity sensor.