Voyager Documentation

Release 0.5

Julio César Castellanos

October 13, 2016

Contents

En sitio estara la documentacion, software y recursos audiovisuales asociados al Proyecto Voyager realizado en el curso de ingenieria del Software I y II.

Voyager is an Arduino bot controlling from a web browser using Nodejs on Docker for Raspberry pi

CHAPTER 1

Presentacion

CHAPTER 2

Contenidos

Los contenidos estan organizados desde un pequeña introduccion y justificacion del porque de este proyecto, seguido de una analisis de requerimientos formal para asi obtener la mejor arquitectura y diseño posible de tanto el software como el hardware que cumpla con el objetivo de este proyecto.

2.1 Introduccion

Technology is just one of many disruptive inluences in education today. We live in an era where the wealth of data and the exponential growth in the development of new knowledge is challenging institutions to rethink teaching and learning in a global market. - Cisco (2013)

As more people adopt new technologies for learning, they will thrive in the emerging world of the Internet of Everything (IoE)—the networked connection of people, process, data, and things—which is becoming the basis for the Internet of Learning Things

2.2 Objetivos

2.2.1 General

- Desarrollar un apliacion (WEB) que permita manipular un microcontrolador arduido y raspberry por medio de una interfaz de facil manejo y compatible con varios modelos de estos.

2.2.2 Especificos

- Prototipar hardware utilizando Arduino que permita comunicación serial en tiempo real utilizando el protocolo firmata y lectura de sensores.
- Codificar una aplicación que reciba peticiones HTTPvia web socket y las transforme a señales digitales utilizando Nodejs con la libreria Jhonny five.
- Desplegar la aplicación en un container de Linux(LXC) compatible con ARM utilizando Docker bajo una Raspberry pi B+ 2 que permita comunicación serial con un Arduino e integre periféricos.
- Diseñar una interfaz web que capte los eventos del navegador para controlar el robot y presente información en streaming de sensores y/o periféricos.

2.3 Justificacion

¿ Cómo construir un prototipo de un robot que pueda ser controlado desde Internet utilizando hardware de bajo coste que apoye la creciente necesidad de aprendizaje en el campo del Internet de las cosas* en las universidades?

La tecnología avanza mucha mas rápido que lo que un programa de universidad puede, en el caso del software es mucho más evidente este avance, pero en lo últimos 2 años el Hardware ha sido protagonista con el surgimiento del movimiento Open Hardware y su enfoque académico con las plataformas como Arduino, Raspberry Pi, Intel Edison. Etc. Además en el mundo de la industria logística y los gatgets la internet de las cosas es tendencia y se estima que para 2017 el mercado sea ade aprox 17 Billones de dolares.

Por esta razón se hace necesario empezar proyectos de desarrollo de software enfocados al control del hardware, los sistemas tradicionales transaccionales han estado por años, ahora es el turno donde los sistemas pueden producir datos y responder en tiempo real a las demandas actuales de soluciones eficientes para ciudades, personas, hogares, empresas.

Lo anterior se justifica bajo las siguientes premisas;

- Micro-controladores y sensores pequeños y baratos
- Nuevas herramientas de software
- La comunidad open source/hardware ha estado bastante activa los últimos años.
- La Internet de las cosas es tendencia

2.4 Requerimientos

2.4.1 1. Configuracion

1.1. Los parámetros se deberán ajustar. El sistema deberá permitir configurar los parámetros de el respectivo microcontrolador.

1.2. Debe restringirse el acceso al control remote de robot con un usuario y una contraseña via web

1.3. Debe existir un sito de configuración para definir lo puertos web, usb, direccion ip, configuracion de interfaz de red.

1.4. Listar direcciones MAC de los dipositivos utlizados

2.4.2 2. Funciones y control

2.1. Los movimientos deben poder controlarse desde cualquier parte. Estos deberán poder controlarse desde cualquier lugar, por medio de eventos del navegador via internet.

2.2. Estado de la batería del dispotivio

2.3. Streaming de video en una interfaz web.

2.4. Control del dispositivo desde el navegar utilizando eventos del teclado, movimiento (izquierda, derecha, adelante, atrás), movimiento de la camara 360.

2.5. Motricidad en cualquier direccion. Generara movimientos en cualquier direccion usando las teclas de direccion de los equipos.

2.6. Si existe mas de un usuario controlador, mostrar el número de conexiones.

2.7. Almacenar en una base de datos el recorrido del robot.

2.4.3 3. Altertas y notificaciones, referecias

3.1. Debe producirse una alerta cuando se pierda la conexión o la batería esté baja.

3.2. Debe permitir configurar alertas basado en lecturas de sensores (proximidad, temperatura, humedad etc.)

3.3. Los elementos del arduino deberán ser referenciados. El sistema deberá visualizar la respectiva referencia de cada componente Arduino.

3.4. Los componentes del Rapsberry Pi serán referenciados. El sistema deberá visualizar la correspondiente referencia de cada componente rapsberry.

2.5 Analisis

2.6 Arquitectura

2.7 Diseño

2.8 Mercury robot challenge 2016

Voyager Technical Document

Universidad Cooperativa de Colombia

Sede Monteria

www.secbi.co

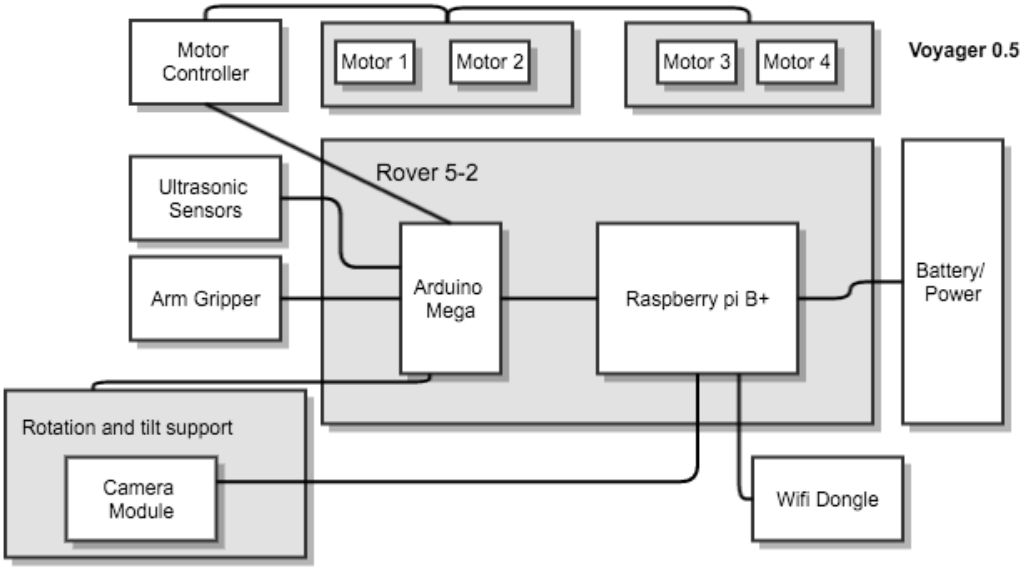
Team

- Julio Cesar Castellanos - julio.castellanosa@campusucc.edu.co
- Ivan Dario - ivan.ortizp@campusucc.edu.co
- Rafael Vergara - Rafael.vergarah@campusucc.edu.co
- Luis Villadiego - luis.villadiego@campusucc.edu.co

2.8.1 Summary

This technical document describes the creation and specs of a remotely controlled robot. The Robot "Voyager" was designed and built to meet the requirements of the Oklahoma State University Mercury Robotics competition for 2016 in Bogota, Colombia.

2.8.2 Highlevel block diagram



2.8.3 Communication

The microcontroller will be conecte via serial to the raspberry pi, on the raspberry we use a Nodejs library in order to expose a real time socket web based server. For handle browser events we use jQuery and keypress in order to detect when a key is pressed then and do a socket emit.

2.8.4 Main board

The robot uses the Raspberri pi B+ as the principal board, this board allow the commnincation to a higer level user application using a Sockect TCP server. It is attached direclty to the microcontroller via serial USB port.

2.8.5 Video feedback

The robot has a camera module attached to the raspberry pi, we expose this camera thougth a UDP video stream server.

2.8.6 Cotroller interface

We have a web based interface using HTML,CSS and Javascript to handle browser events.

2.8.7 Drive train

The robot has one motor per side, each running a tread through a gearbox with an encoder.

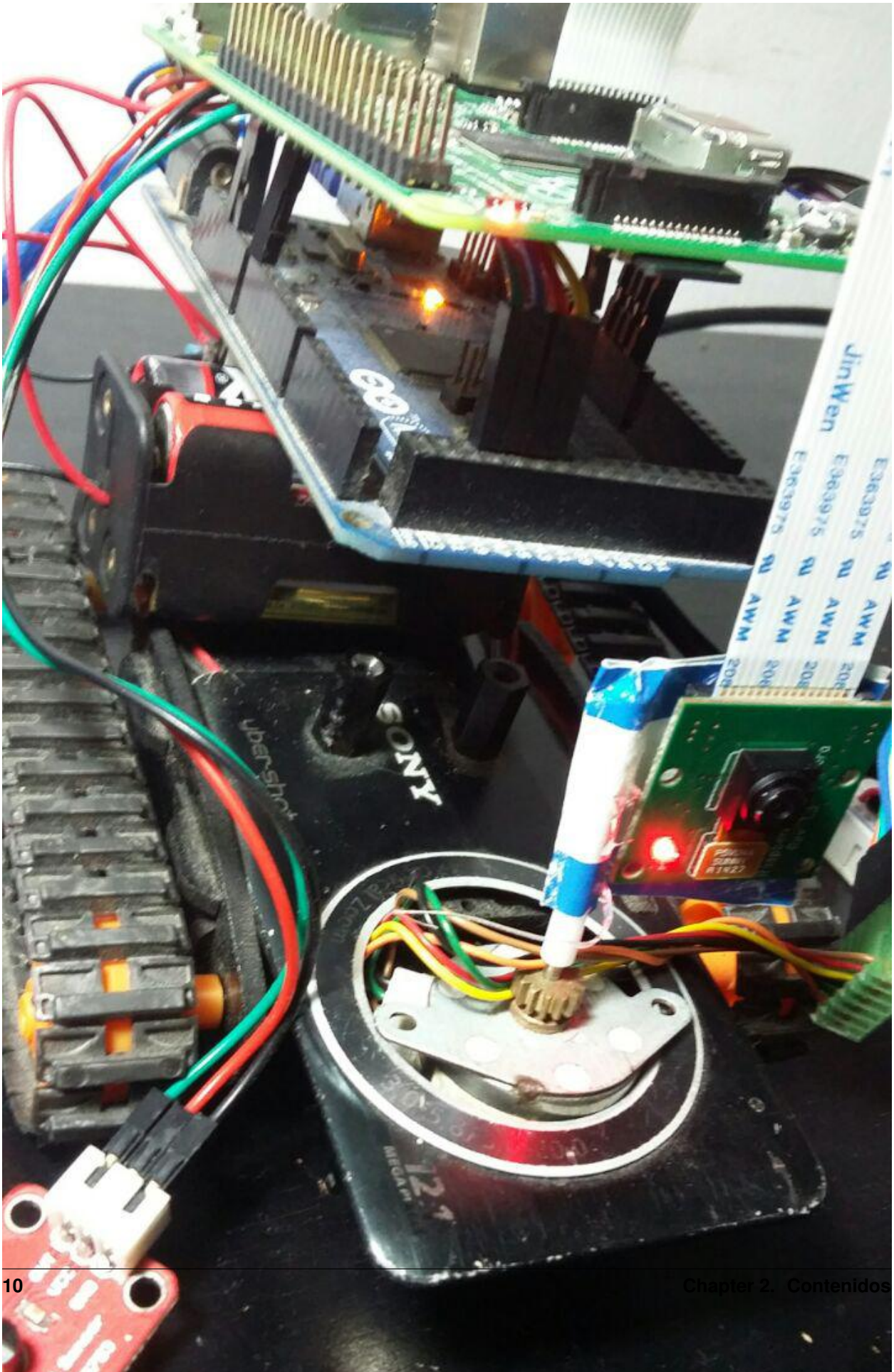
2.8.8 Subsystems

The robot has two Ultrasonic sensors.....

2.8.9 Power

The robot has a 7.2[V] standard 3000[mAh] NiMH battery.

Note: La foto acontinuacion corresponde el primer prototipo con una placa de Ardunio MEGA 2560, una Raspberry Pi B+ con su camara, un sensor LM35 y puente H LN28N. La Rpi se conecta a la Wifi y se comunica con el Arduino via serial.


JinWen
SONY

CHAPTER 3

Configuracion inicial

Para empezar a desarrollar con este proyecto es neserio realizar una configuracion incial, y la conexion de los dispositivos de hardware.

Acontinuacion la lista de dispositivos a utilizar y las dependencias de software

CHAPTER 4

Hardware

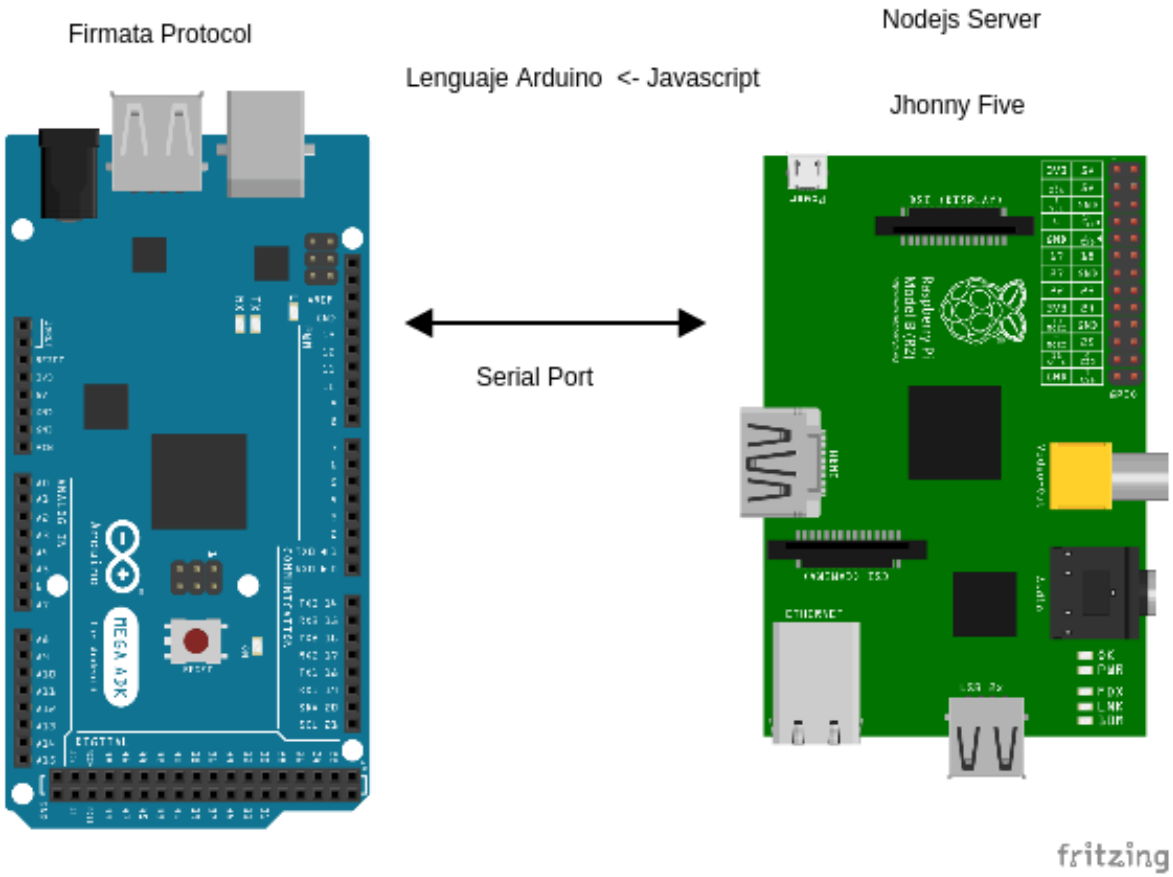
- Raspberry pi
  - Camera board
  - Usb Wifi dongle
- Arduino
  - DC Motors
  - H bridge
  - Sensors

CHAPTER 5

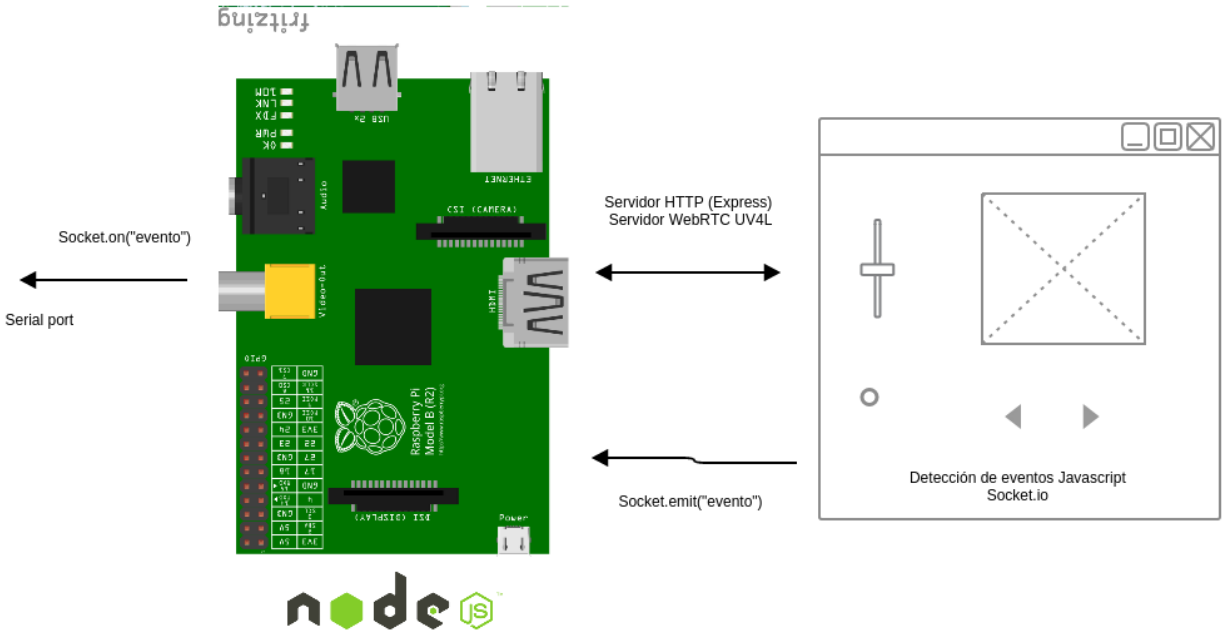
Arquitectura

Asi sera la conexion basica entre el Ardunio y la Raspberry pi, y la interaccion de las partes de software

En la *Fig.1* podemos observar que la forma de comunicacion entre la reaspberry y el ardunio es via serial, utlizando una libreria de Nodejs llamada *serial port*.



En la *Fig.2* se utliiza un servidor HTTP basico de Express con Socket.io para escuchar eventos en el navegar en tiempo real utilizando Javascript.


Socket.on("evento")
Serial port
Servidor HTTP (Express)
Servidor WebRTC UV4L
Socket.emit("evento")
Detección de eventos Javascript
Socket.io
Raspberry Pi
Model B (R2)

CHAPTER 6

Software

- Firmata (Arduino)
- Nodejs
  - Jhonny-five
  - Socket.io

CHAPTER 7

Primeros pasos

Lo primero que debemos hacer es hacerle un "flash" a la targeta de Ardunio con el protocolo *Firmata*.

Para este experimeinto usamos una version modificada de Firmaata que funciona con sensores de proxomidad servor motor.

En `src/sketch.ino` esta listo el archivo para subir a la placa. Para hacer esto utlizamos la herramienta de linea de comandos `ino`

7.1 Instalar el Ardunio IDE

Lo podemos instalar con `yum` o `apt-get`.

```
$ yum install arduino
```

7.2 Instalar picocom

La herramienta de linea de comandos `picocom` nos ayudara con la comunicacion serial desde la Raspberry pi.

```
$ wget https://picocom.googlecode.com/files/picocom-1.7.tar.gz
$ tar -xvzf picocom-1.7.tar.gz
$ cd picocom-1.7
$ sudo make
$ sudo make install
```

Una vez instaladas las depencias podemos ahora instalar `ino` usando `pip` o `easy_install`

```
$ pip install ino
```

7.3 Configurando ino

Editamos el archivo `ino.ini` con las especificaciones de la placa de Ardunio y los puertos que usa.

```
[build]
board-model = mega2560

[upload]
board-model = mega2560
```

```
serial-port = /dev/ttyACM0

[serial]
serial-port = /dev/ttyACM0
```

7.4 Instalando el Protocolo Firmata

Compilamos el archivo en `src/sketch.ino` utlizando ino.

```
$ ino build
```

y lo subimos a la placa.

```
$ ino upload
```

7.5 Instalando las dependencias NPM

Ahora que la placa de Arduino esta lista, podemos utulizar Nodejs con Jhonny-five.

Debemos asegurarnos que este instalado Nodejs y npm en la Raspberry pi.

```
$ npm install
```

Luego que termine la instalacion, conectamos el Ardunio a la Raspberry pi, corremos la app:

```
$ node app.js
```

Observaremos lo siguiente:

7.6 Servidor y cliente

Configuramos un servidor HTTP en `app.js`

```
app.listen(8000, function () {
    console.log('Http server listening on port %d', 8000);
});
```

Con eso podremos acceder desde el navegador al cliente Socket.

Para manejar eventos utilizamos jQuery y keypress con la finalidad de detectar cuando una tecla es presionada y hacer un `socket emit`, asi:

```
"keys": "up",
"on_keydown": function() {
    console.log("Client: Going forward");
    socket.emit('goForward');
```

Del lado del servidor escuchamos este emit utilizando `socket on`

```
socket.on('goForward', function(){
    console.log("Server: Going forward! ");
    // Do something
});
```

7.7 Running in a Docker container

Los contenedores son una muy buena idea para los proyectos de la Iot porque podemos aislar a nuestra aplicación en un nivel de kernel y eso significa portabilidad en cualquier plataforma que soporte Docker, esto garantiza un despliegue rápido de aplicaciones, algo que es crucial para proyectos de Iot.

Pero Docker, que es un estándar de facto, no es compatible con ARM. Asi que aun no es complatible con Raspberry Pi, afortunadamente, algunos hackers han modificado una Distro De Raspberry Pi, que llamaron Hypriot.

Para esto instalamos Hypriot de esta *guia*

Nos logeamos a la Rasberry pi.

Clonamos el repo

```
$ git clone https://github.com/juliocesar-io/voyager-bot.git
```

```
cd voyager-bot
```

Compilamos la imagen

```
$ docker build -t <your_tag_name> .
```

Conectamos el Ardunio(Con Firmata) a la Raspberry pi.

Ejecutamos el contenedor.

```
$ docker run --device=/dev/ttyACM0 <your_tag_name>
```

Es importante definir buen el puerto con el flag `--device=/dev/ttyACM0`, si no sabemos el puerto podemos consultarlo con `lsusb`.

Vamos al navegador y probamos que todo funcione en `http://<container_ip>:3000`