ATmega328P como IoT

Una implementación de control basada en web.

El presente proyecto tiene como intención el aprovechamiento de tecnologías web para la manipulación de equipos, máquinas y la obtención de información, de forma remota agnóstica del dispositivo y sistema operativo empleados.

Introducción

Esta experiencia consta de dos partes:

- Un ATmega328P programado con un intérprete de comandos que recibe ordenes via serial (UART), montado en una placa Arduino.
- Un servidor web programado utilizando NodeJs, Express y socket.io, que provee una interfaz gráfica, contra el intérprete instalado en el microcontrolador.

Componentes:

- Placa Arduino Uno rev3	x1
- Motor PaP Bipolar	x1
- Driver tipo Pololu a4988	x1
- Pulsadores/Fines de Carrera	x2
- Led	x1
- Resistencia 220 ohm	x1 (más una por c/pulsador)
- Protoboard	x1
- Capacitor polarizado 100uF	x1
- Cables	Varios

Diagrama de Flujo:

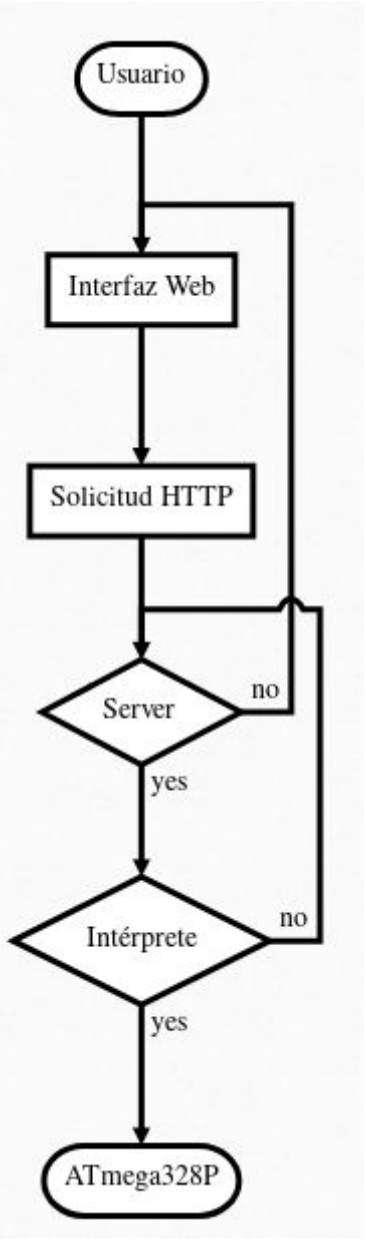
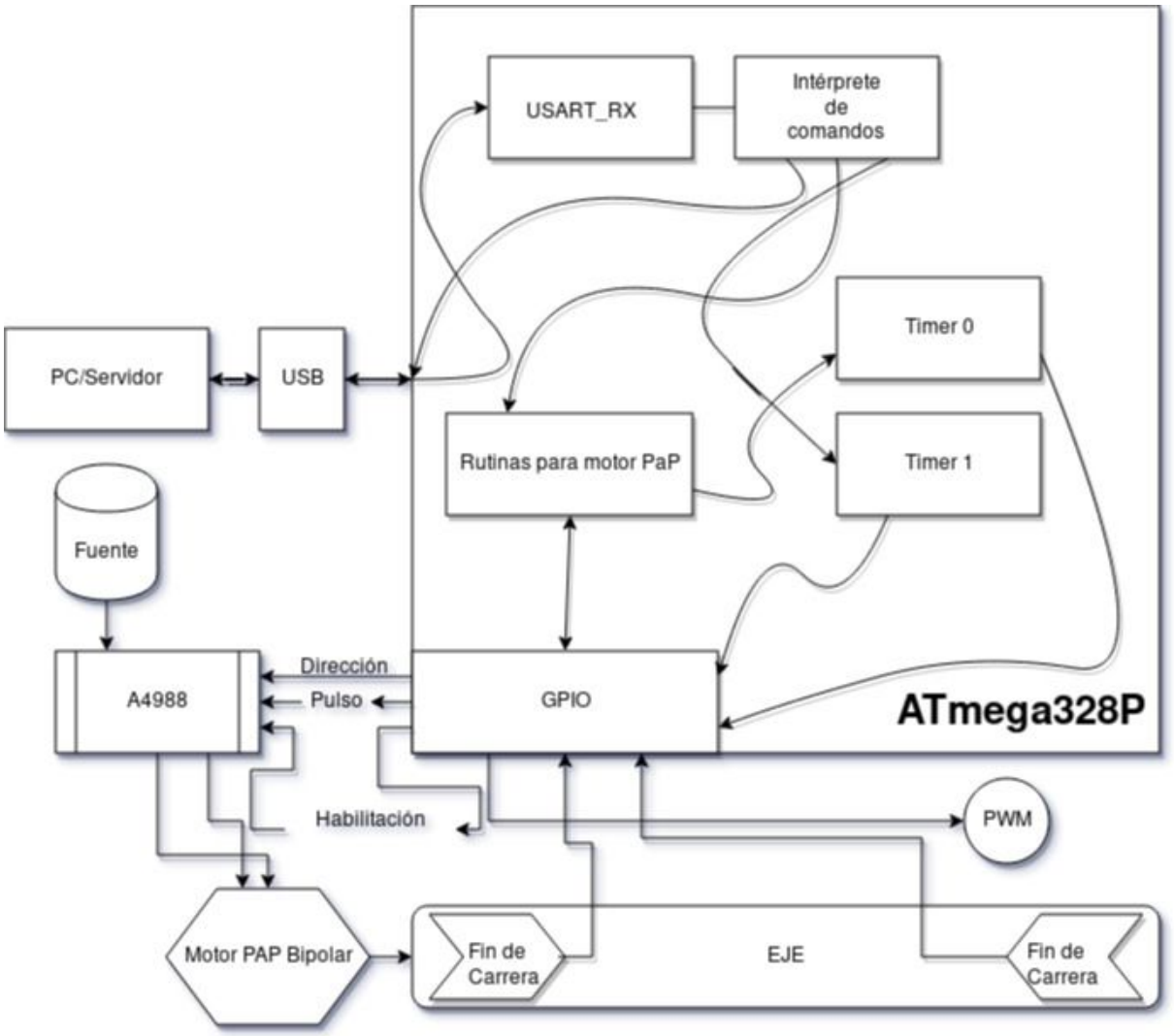


Diagrama de Bloques:



Descripción General:

El dispositivo opera de la siguiente forma:

1. Se Inicia el servidor, el cual se conecta con el ATmega328P por el puerto serie, y comienza a servir una página web estática en el puerto 8080.
2. El Usuario interactúa con la Interfaz web específicamente desarrollada para esta aplicación. La página envía los comandos como solicitudes http al servidor que la esté albergando. Además cuenta con una casilla de mensajes que permite recibir mensajes del mismo, esta le permite:
   - Iniciar la rutina de referenciación del eje (homing)[1].
   - Obtener diagnósticos de posición y velocidad instantáneas en tiempo real[1]
   - Indicar gráficamente la posición deseada el motor[1]

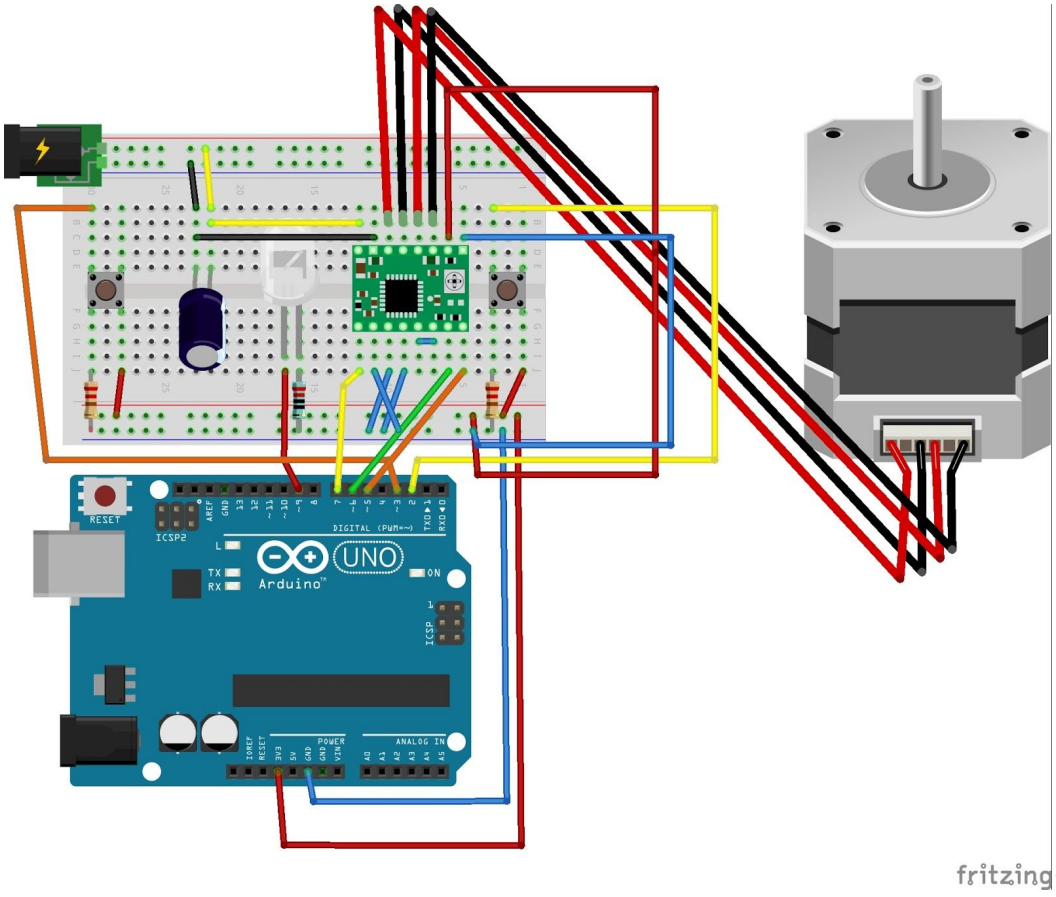
[1] Todos los comandos se envían al servidor como solicitudes http

- Establecer gráficamente la velocidad del motor (en RPM)[1]
- Recibir mensajes provenientes del microcontrolador.[2]

3. Los comandos provenientes de la página, como solicitudes http, son procesados y traducidos a órdenes que el microcontrolador pueda interpretar. Estas son enviadas por el puerto serie. Cualquier respuesta por parte del intérprete es enviada mediante un socket a la página para informar al usuario.

4. Las órdenes que llegan por UART son interpretadas por un intérprete de comando que las ejecutara, devolviendo en caso necesario una respuesta de error o éxito.

Esquema:



Descripción ATmega328P

El Microcontrolador realiza una serie de tareas desde su encendido hasta su disposición a recibir comandos.

1) Establece el stream por defecto (salida de printf, etc) a través de las funciones personalizadas para la comunicación por UART.
2) Se inicializa la UART, y se la enlaza con el intérprete de comandos.
3) Crea una nueva instancia de motor la cual:

[2] Los mensajes se reciben mediante un socket enlazado con el servidor
[3] El esquema fue realizado con la iniciativa libre Fritzing

   a) Establece como salidas los pines indicados para dirección, pulsos, y habilitación.
   b) Precalcula los pulsos por vueltas que luego serán utilizados para establecer la velocidad en RPM.
   c) Se carga la instancia en un arreglo global de motores.
4) Se establece como salida el pin 9 OCR1A.
5) Se inicia el timer 1 con un prescaler x8, en modo fast pwm, con los actuadores COM1 A y B en modo clear (proporcional) y una frecuencia de trabajo ~31Hz (para que el pwm sea fácilmente visible).
6) Se habilitan las interrupciones por cambio de pin en los pines conectados a los fines de carrera.
7) Se habilita el timer 0 con un prescaler x8, y se habilita la interrupción por overflow.
8) Posteriormente se habilitan las interrupciones.
9) Se establece el pin 13 como salida (led embebido).
10) Se envía un mensaje indicando que el controlador está listo.
11) Se entra en el loop de trabajo que alterna el estado del led incluido en la placa cada medio segundo.

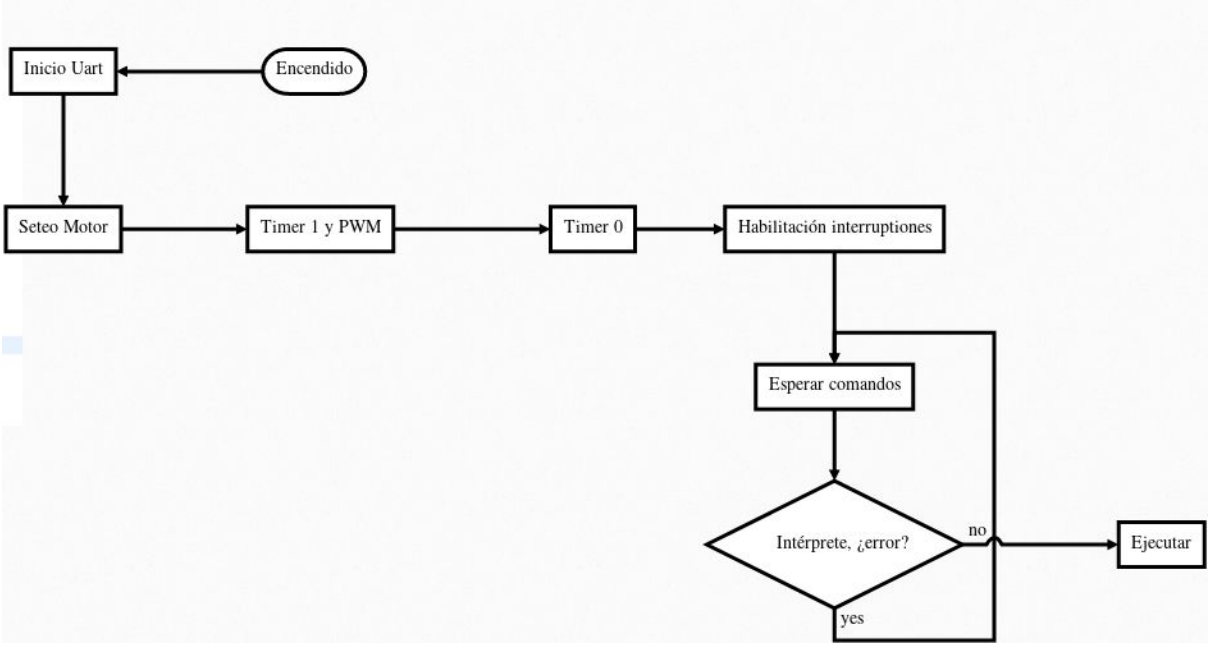
Cuando llega información por la UART se procede de la siguiente forma:

1) Se revisa el caracter recibido, si no es un retorno de carro o nueva línea se procede a almacenarlo y a aumentar un contador. En caso contrario se evaluará el buffer almacenado con el intérprete de comandos.
2) El intérprete de comandos evalúa la cadena de caracteres recibida, en caso de que el comando no sea reconocido reporta error. Si el comando incluye argumentos, se los traduce a números. Si el comando se encuentra en un formato incorrecto, reporta error.
3) Si no hubo error se procede a ejecutar el comando y, en caso de ser requerido, reportar información.

Las órdenes disponibles por el intérprete son:

-> :Fn:nnn :Forward<motor>:<stepps>
-> :Bn:nnn:Backward<motor>:<stepps>
-> :Sn:nnn :Speed<motor>:<RPM>
-> :Pn:nnn :Absolute Position<motor>:<position>
-> :Rn:nnn :Relative Position<motor>:<percentage>
-> :H :Homing Routine
-> :Wn :Wait (stops)<motor>
-> :Dn :Diagnostic<motor> Shows current position & speed of selected engine
-> :Qn :Quiet (brake)<motor>
-> :h :Shows this help message

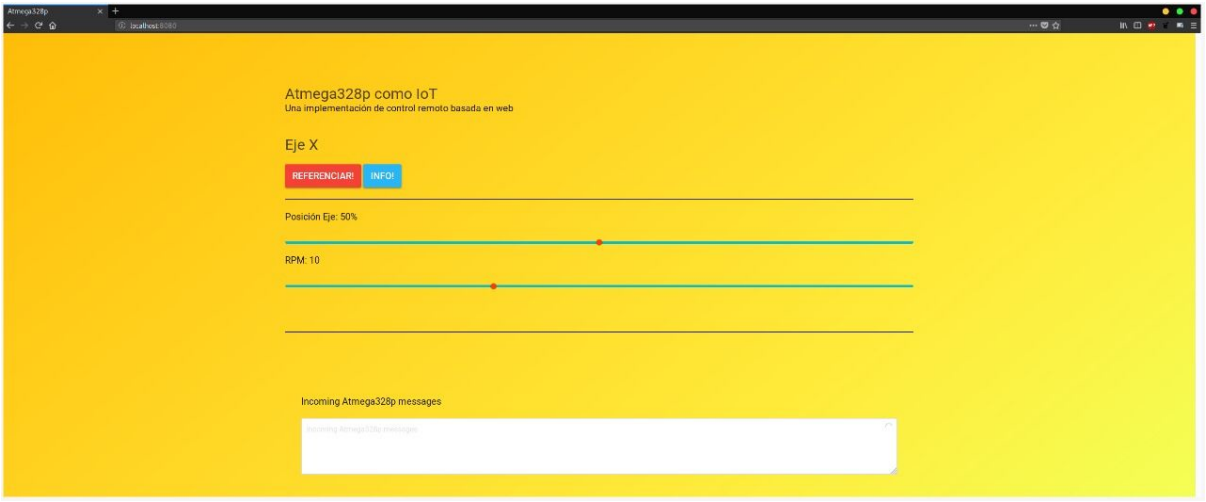
Diagrama de Flujo:



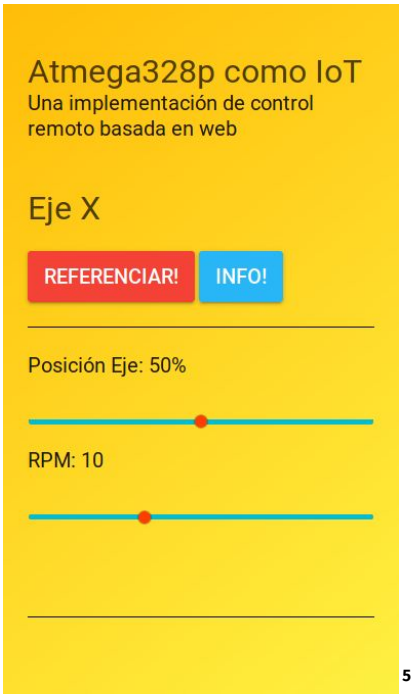
Descripción Servidor:
El servidor fue escrito en nodeJs con ayuda de los framework express y socket.io. Además se utilizo el módulo node-serialport para las comunicaciones seriales.

1. Al arranque el servidor escanea un puerto hardcodeado en busca de una Arduino
    a. Si no lo encuentra reporta error y aborta.
2. Posteriormente Inicializa el servidor web, y sirve la interfaz web en el puerto 8080 del localhost.
3. Ante la recepción de una solicitud por parte de la página, se extrae la información que esta trae, se la traduce a un formato que el intérprete presente en el microcontrolador pueda ejecutar y se la envía al mismo.
4. Espera a la siguiente solicitud.
5. En caso de respuesta por parte del ATmega328P se la envía mediante un socket a la página.

Descripción Interfaz Web:



[4]



[5]

La interfaz cuenta con 3 secciones:

A. Información y Acciones básicas, que a su vez consisten en:
    a. Botón “REFERENCIAR!” -> Inicia la maniobra de Homing
    b. Botón “INFO!” -> Solicita datos al micro.
B. Sliders (disponibles luego del referenciado):
    a. Posición -> ordena al microcontrolador a posicionar el eje en posiciones relativas fácilmente visualizables por este elemento.

[4] Página vista desde un ordenador
[5] Versión para móviles

   b. Posición -> ordena al microcontrolador a posicionar el eje a una velocidad dada dentro del margen permitido por este elemento.

C. Casilla de mensajes:
   a. Aquí se muestran los mensajes provenientes del microcontrolador.

La página reacciona a eventos, como apretar un botón o mover los sliders, enviando solicitudes al servidor con la información del evento.

Conclusiones:

El ATmega328P es un controlador no diseñado para IoT. Sin embargo acompañado de alguna interfaz que provea el medio de comunicación adecuado (en este caso un ordenador comunicado por el puerto serial, o un microcontrolador ESP8266 comunicado por SPI o I2C), puede desempeñarse perfectamente como si lo fuera.

La creación de una interfaz web permite controlar al dispositivo en tiempo real desde cualquier plataforma o aparato, sin necesidad de drivers o instalaciones tediosas y sin limitaciones de distancia. Por ejemplo si el servidor fuese expuesto a la wan, cualquier dispositivo conectado a la misma podría controlarlo desde cualquier lugar del mundo.

El servidor, la interfaz web y el programa cargado en el microcontrolador fueron desarrollados con escalabilidad en mente, lo que resultó en una plataforma fácilmente extensible al control de una cantidad indefinida de motores. Vale aclarar que por limitaciones de conexión, en el caso de la plataforma Arduino Uno que sólo tiene 11 pines digitales disponibles,(0 y 1 están reservados para rx y tx). Está limitado a un número escaso de motores. Si no fuese así, la cantidad podría aumentarse hasta que las demoras por la rutina de interrupción por overflow del timer introdujesen latencias tales que no fuese práctico.

La mayor limitación que inhibe la cantidad de motores a controlar, es entonces, la rutina de manejo. Los motores son controlados por una ISR ante cada overflow del timer 0 (7,8 kHz, preescaler por 8).
Un control más adecuado sería acelerar completamente por hardware la emisión de pulsos mediante las salidas asignadas a los timers presentes en el microcontrolador (control por pwm). Esto permitiría controlar 6 motores de manera eficiente sin restarle tiempo a la rutina principal del integrado (main).

Todo el código del proyecto puede encontrarse en el repositorio
https://github.com/Waaflee/CVACR

Herramientas y librerías utilizadas

- Librería de HAL AVRDuino de mi autoría https://github.com/Waaflee/AVRDuino
- Librería para controladores avr avr-libc https://www.nongnu.org/avr-libc/
- Compilador para mmcu de ATmel https://gcc.gnu.org/wiki/avr-gcc
- Utilidad para subir programas a la mmcu https://www.nongnu.org/avrdude/
- Framework para comunciacion en tiempo real servidor-página https://socket.io/

- Runtime de javascript para el servidor https://nodejs.org/en/
- Módulo para la comunicación NodeJs<->Serial https://github.com/node-serialport/node-serialport
- Framework para la programación de servidores web https://expressjs.com/
- Cliente HTTP https://github.com/axios/axios
- Framework CSS https://bulma.io/
- Utilidad para creación de diragramas de flujo http://flowchart.js.org/
- Software libre para la realización del esquema del circuito www.fritzing.org