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

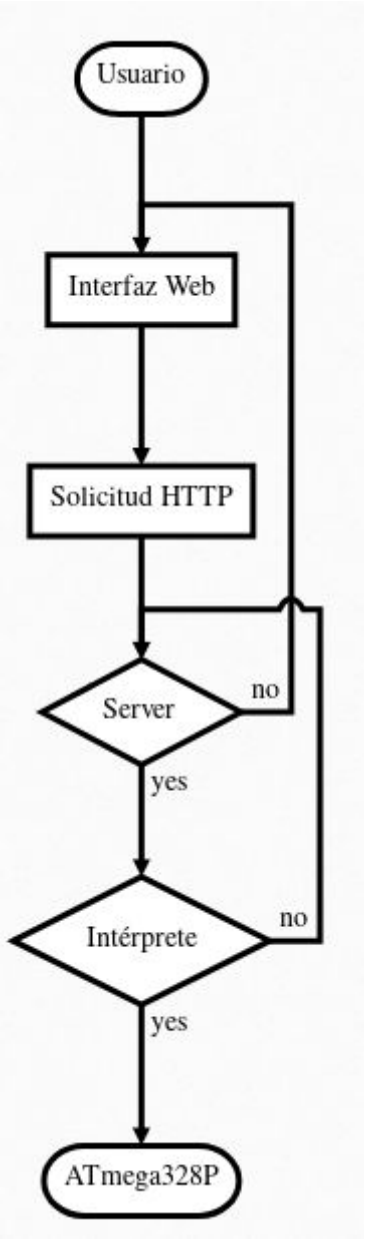
Esta experiencia consta de dos partes:

- 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:



Descripción:

El dispositivo opera de la siguiente forma:

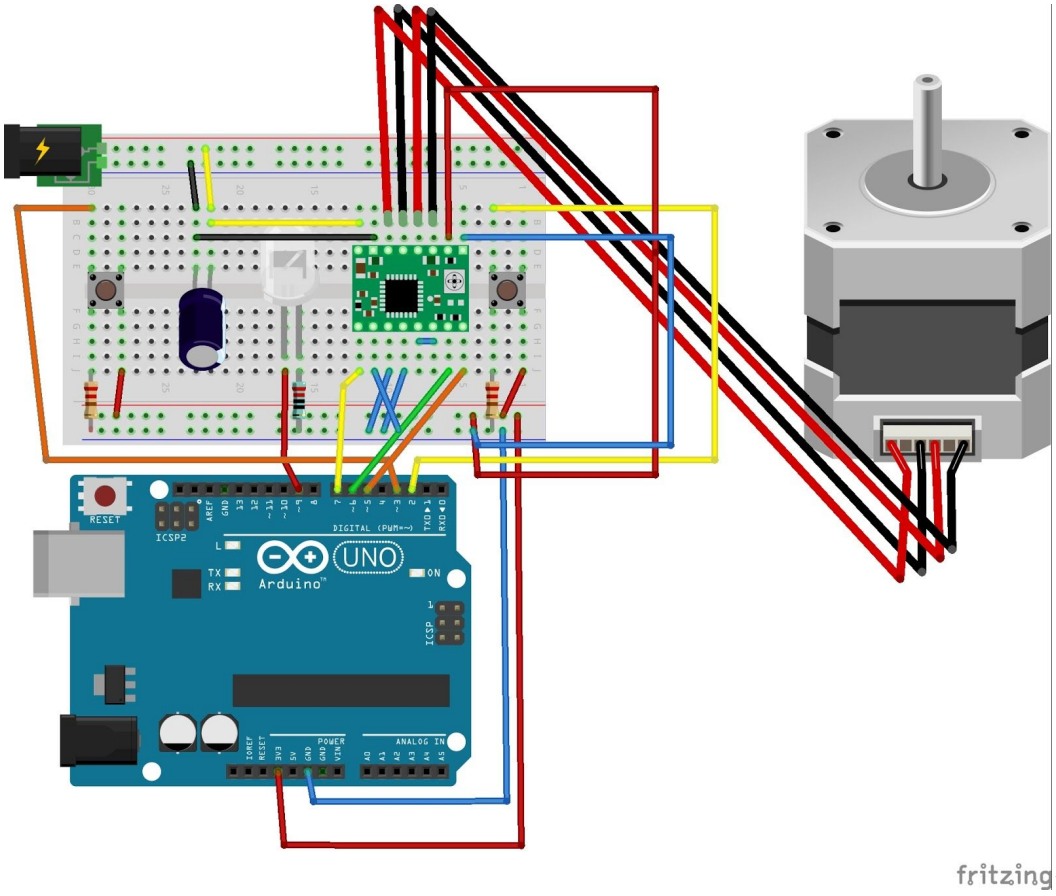
1. Se Inicia el servidor, el cual se conecta con el ATmega328P por el puerto serie, y comienza a servir una página web estática en el puerto 8080.

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]
   - 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 traducido 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:



[1] Todos los comandos se envían al servidor como solicitudes http
[2] Los mensajes se reciben mediante un socket enlazado con el servidor
[3] El esquema fue realizado con la iniciativa libre Fritzing

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/
- Framework para la programación de servidores web https://expressjs.com/
- 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