SN173 Demonstration: Iron Man!

The SN173 is a versatile prototyping board built to provide full access to the capabilities of the SM220 SNAP module. This demo application uses SN173 boards to showcase:

- Interfacing to an HC-SR04 Ultrasonic Sensor
  - Low cost sensor for distance measurements
  - Shows precise input pulse timing using SM220 “input capture” hardware
  - Wireless sensor reporting using SNAP RPC calls
- Interfacing to standard R/C servos
  - Versatile, low cost, and readily available – SM220 can drive up to 8 servos directly with hardware PWM
  - Shows generating precision one-shot and PWM pulses with the SM220
- Intelligent embedded control using networked sensors
  - Programming an “animatronics” control-loop with SNAP
- Connecting an Internet gateway to a sensor network
  - SNAP Connect python library
  - HTML5 websocket based user interface

Elements of the Demo:
These parts “Assemble” to make the Iron Man demo, but they are very usable in standalone form for a diverse range of projects.

- Ultrasonic Sensor Interface
  - SNAPpy script: sonic_ranger.py
  - HC-SR04 interface circuit for SN173
- Servo Driver Interface
  - SNAPpy script: pan_tilt.py
  - Dual servo interface circuit for SN173
- Embedded Intelligence – animatronics
  - SNAPpy script: iron_head.py
- Web Application
  - SNAP Connect program: web_app.py

Ultrasonic Sensor Interface

Power
The HC-SR04 requires 5v to operate, so we can't supply it directly from the SN173's 3.3v VCC output. Instead, we connect the HC-SR04 to the SN173's 5V line. By providing an external power input on the adapter board, we have a choice of powering the whole assembly from a battery-pack or from the USB input.

Signals
A short pulse (nominally 10us) on the Trigger input initiates an ultrasonic measurement cycle. The Echo output goes high for a duration equal to the elapsed time between ultrasonic send and receive. Calculating the distance the sound traveled is a simple matter of multiplying this elapsed time by the speed of sound.

Interfacing between 5v and 3.3v
Care must be taken when interfacing signal lines between the SM220 which we'll be operating at 3.3v, and this 5v peripheral. The SM220's internal I/O is diode-clamped to offer some protection from over-voltage, but we want to ensure that only a small current flows through those diodes. This may not offer enough protection for production circuits – consider using purpose-built level shifters for increased reliability. However, the design goal for this demo is "easy to hand-wire". We connect the Trigger output from the SM220 directly to the HC-SR04 input, relying on a high-impedance input circuit to limit the current. For the Echo input to the SM220, we use a 1k series resistor for current limiting.

Measuring the Echo Pulse
The SM220's internal CPU (ATmega128RFA1) has two "input capture" capable I/O pins. These are configured in the demo script – one to capture the timestamp of the rising-edge, and the other to capture the falling-edge. Both inputs are tied together and connected through the current-limiting resistor to the Echo output. After sending the Trigger pulse, we can just subtract the falling from the rising timestamp to get the precise time "in flight".

Servo Interface

Common R/C servos have 3-wires: Signal, Power, and Ground. The SN173's VCC (3.3v) is not sufficient for many servos, so we will provide an external power input on the adapter board. This gives us a choice of powering the whole assembly from a battery-pack or from the USB input. It also provides some isolation between the servo power rail and the SM220, which is a good thing due to motor electrical noise and transients.

Controlling the Servo Position
Servos are controlled using "pulse position modulation", with a pulse-width of 1.5ms driving the servo arm to center position. Pulse-widths ranging from 1-2ms will typically drive the servo over its nominal range of motion. Pulses must be sent continuously to hold the servo position, nominally at a 50Hz rate. The pulse rate does not control the position, only the pulse-width.

Servos can be controlled using a software-only approach, for example using pulsePin(-1500) to deliver a 1.5ms centering pulse from within the 10ms timer tick. However, greater accuracy and far less CPU load is gained by using the dedicated timer/PWM hardware in the SM220. Two PWM pins are used in the Dual Servo Interface circuit.

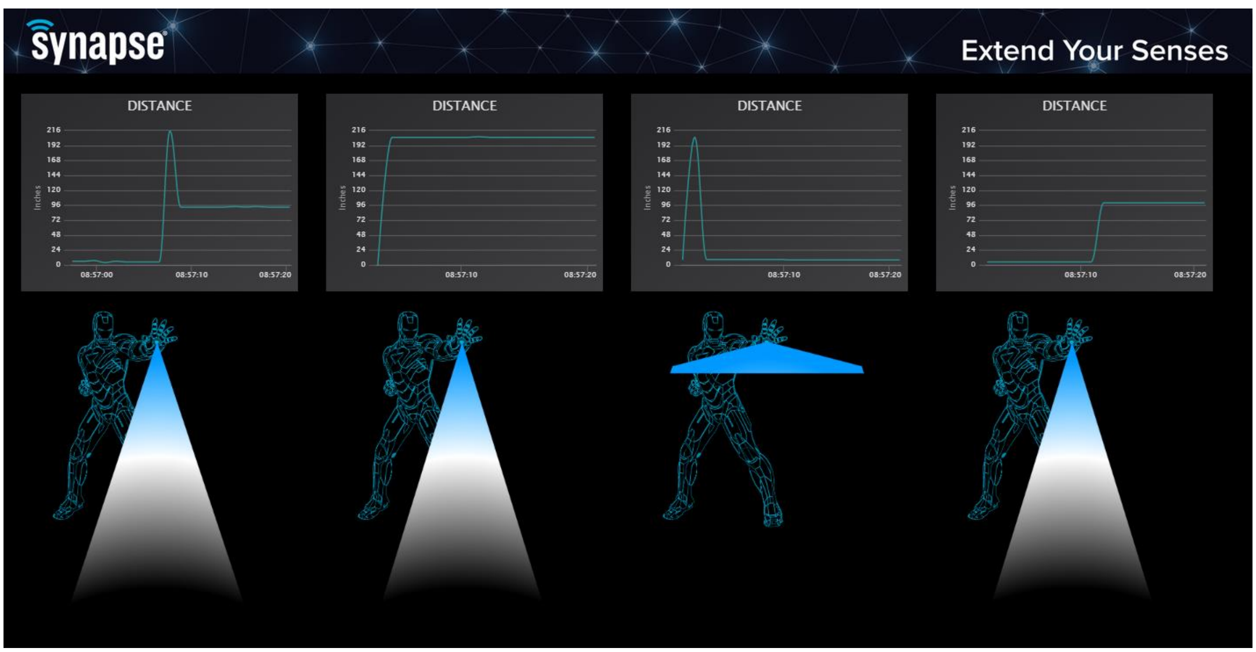
Embedded Intelligence – Animatronics!

Often the best way to increase the performance of a networked application is to move the intelligence to the edge of the network. Processing can be done, decisions made, and actions taken without requiring a roundtrip to the gateway or cloud. SNAP enables this by providing dynamic programmability at the edge, extending the application programming to IoT devices rather than treating them in a more traditional "fixed function" way.

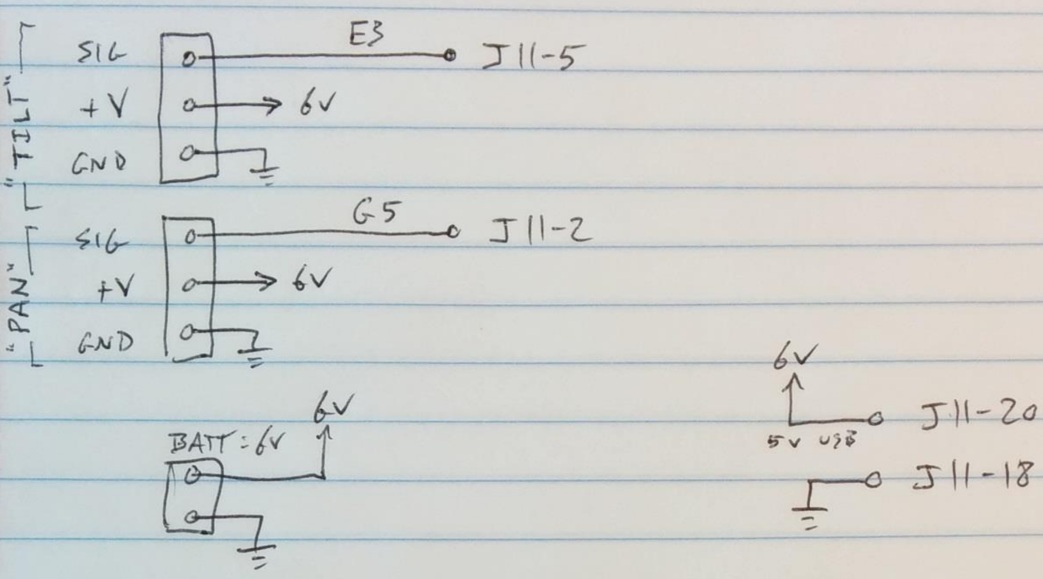
In our demo application, an Iron Man head is attached to a pan/tilt assembly controlled by two R/C servos. The embedded control logic in iron_head.py reacts autonomously to "distance" events transmitted by the sonic_ranger.py nodes. The effect is that Iron Man turns his head to look at the nearest object based on sensor reports.

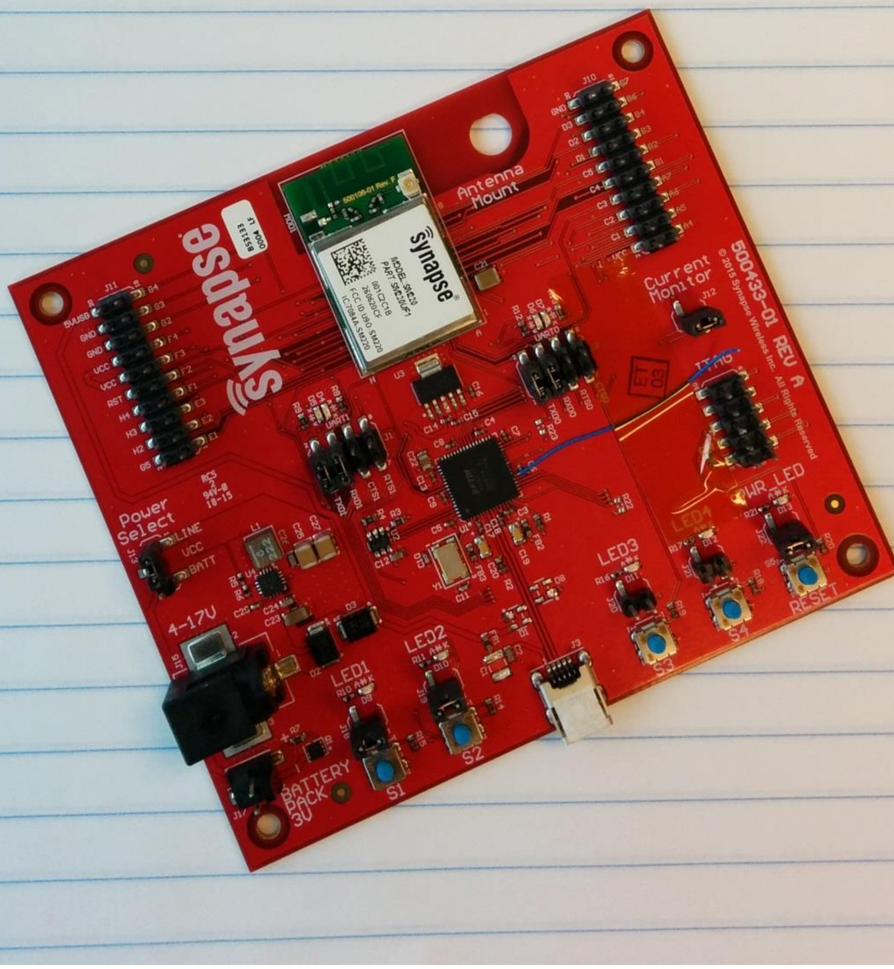
Web Application

In addition to the embedded devices, the demo contains an HTML5 client which displays the status of all sensors. The program app_server.py is a web server based on Tornado, integrated with SNAP Connect.



Servo Controller





Ultrasonic Ranger