TapIn Cloud Based ARDUINO Control

TAP IN SYSTEMS

Web site: www.tapinsystems.com

Email: info@tapinsystems.com

Contents

License

Resources

It is presumed that the reader is already familiar with the Arduino Mega 1280 and 2850 architecture and is also familiar with the Wiring programming system. If you are not familiar with the Arduino Mega, go here to start your journey: http://arduino.cc/en/Tutorial/HomePage

Overview

Using Tap In System's CPX system and TapIn service, you can create Cloud-based control processes, called TAPs, for your Arduino controller. TapIn allows you to build and control your own custom Internet of Things (IoT) device without any cloud programming using a graphical, web-based interface. This capability enables the construction of Arduino-based hardware that is cloud accessible, and can work with with other cloud-accessible devices and services (such as Phillip's Hue, Twitter, Nest, Dropbox, etc.) within a common, integrated process model.

The following steps are required to build a CPX-based IoT system.

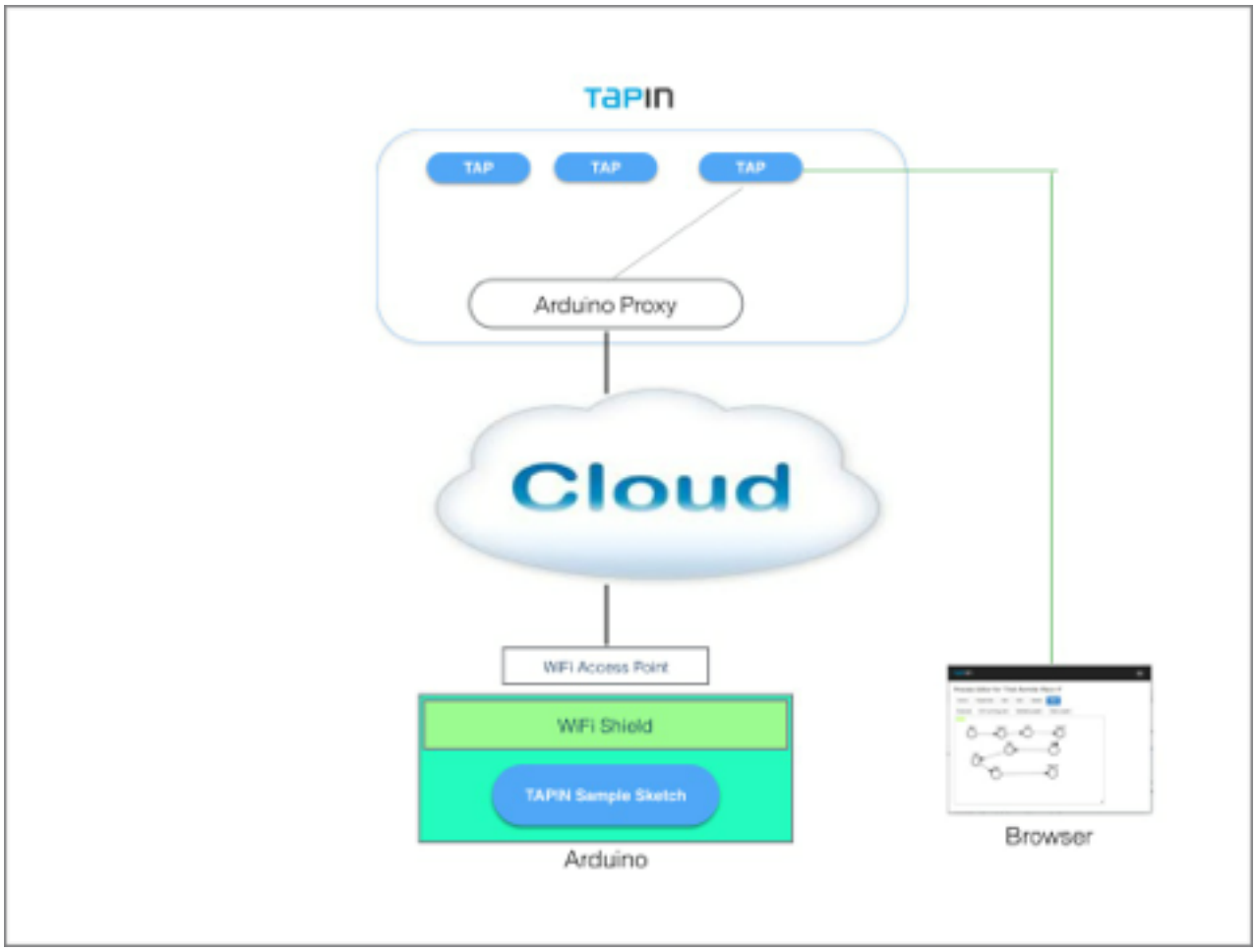
- Set up your Arduino project.
- Download the CPX program for the Arduino from Tap In System's git repository.
- Obtain a TapIn service account (the service is free).
- Install the sample code on your Arduino.
- On the TapIn service, copy the sample Arduino TAP into your account.
- Execute the TAP from the TapIn service to demonstrate cloud control of your Arduino.
- Extend the functionality to match your own Arduino project requirements by modifying the TAP and/or sample Arduino code as necessary.

The implementation section describes each step in greater detail.

System Architecture

The TapIn Arduino architecture is shown in the following diagram.

TAPIN- ARDUINO CONFIGURATION



The main components are:

1. TapIn Service. This is a cloud-based service on the Internet at www.tapinsystems.com. Any user can register for this free service. Users have access to TapIn Action Plans, or TAPs, that are executable process models that control and integrate web-enabled services. TapIn provides sample TAPs that control Arduino devices.
2. TapIn Arduino Cloud Proxy. This is a sub-component of the TapIn service which serves as a proxy between the TAPs and each users Arduino. When a user's Arduino starts, it registers

with the proxy. When a TAP needs to execute a command to a device, communicates with the proxy, which forwards the command to the appropriate device.

3. Arduino Hardware. This is your Arduino project. It can consist of any Arduino add-on component. The only TapIn requirement is that it connects to the Internet. The basic TapIn Arduino CPX assumes that a Wifi shield is used.
4. TapIn CPX Arduino code. This code is loaded on the Arduino via the Arduino IDE as a sketch. It supports the CPX protocol (described later) between the TapIn proxy and the Arduino hardware. This code is distributed by Tap In Systems as open source at https://github.com/Tap-In/CPX. The CPX example shows controls for LEDs, an RGB lamp and a button. This code can be extended to support other components. Additional samples showing control for other Arduino components will be provided by TapIn in the future.
5. Wifi access point. The Arduino WiFi configuration in our sample requires a local access point to be defined.
6. Web browser to access the TapIn cloud service. TAPs can be created and controlled from a standard web browser.

Implementing the TapIn Arduino sample

The following are the detailed steps for implementing the TapIn service using the TapIn CPX Arduino code.

1. Set up your Arduino project.

   A. Download the Arduino IDE from http://www.arduino.cc.

   B. Assemble your Arduino hardware. This example uses the Arduino Mega 2850. The TapIn-CPX code assumes a wifi shield is installed for Internet connectivity.

   C. Install the appropriate libraries on your IDE to support your hardware. Run the test programs to ensure your hardware is operational.

2. Download the CPX program for the Arduino from Tap In System's git repository at https://github.com/Tap-In/CPX.

   A. Install the CPX library on the Arduino IDE by importing the CPX library.

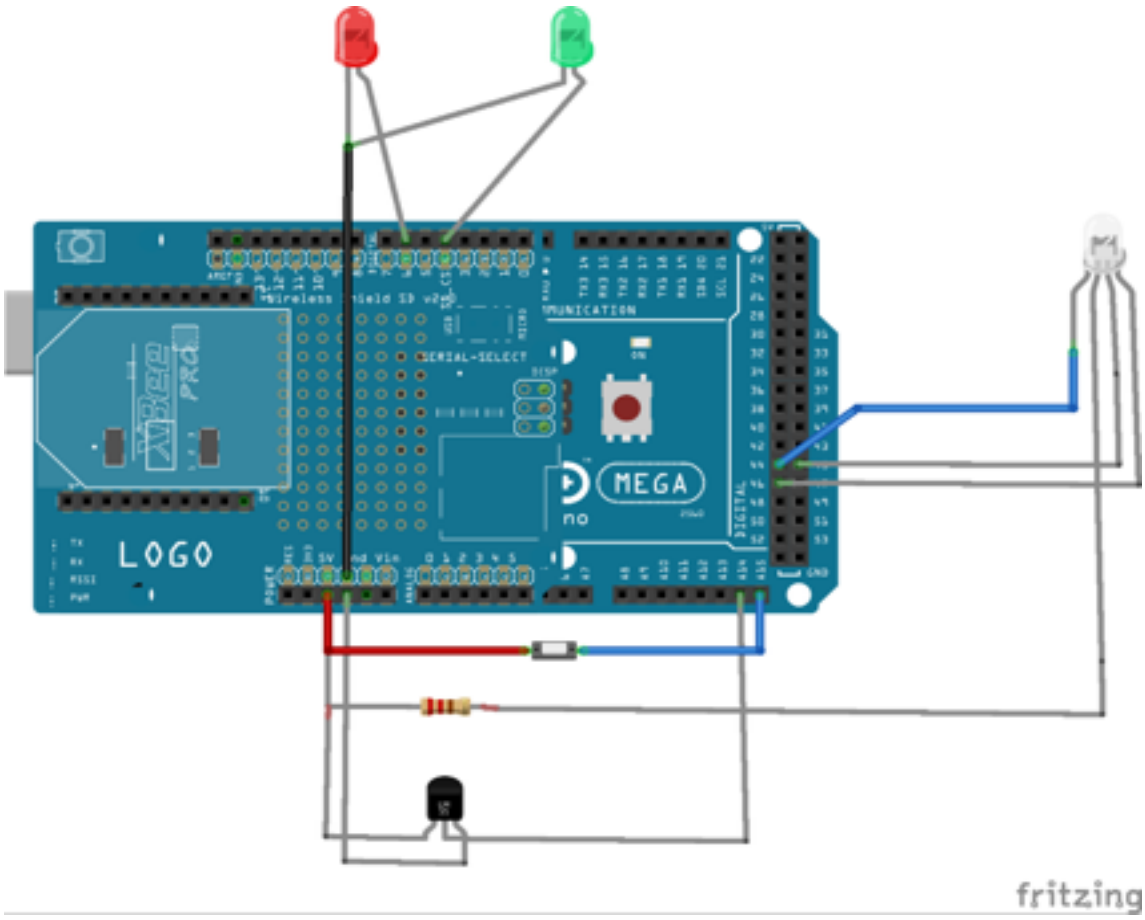
   B. Open the CPX example named Doohickey with your IDE. Modify the pin connections as required to meet you Arduino configuration. The pin configuration and the wiring diagram is shown below. See the documentation in the source download for the exact Doohickey configuration.

      i. Green LED - pin 4

      ii. Red LED - pin 6

      iii. RGB Lamp - pins 44, 45, 46.

      iv. Button contact - pin A15



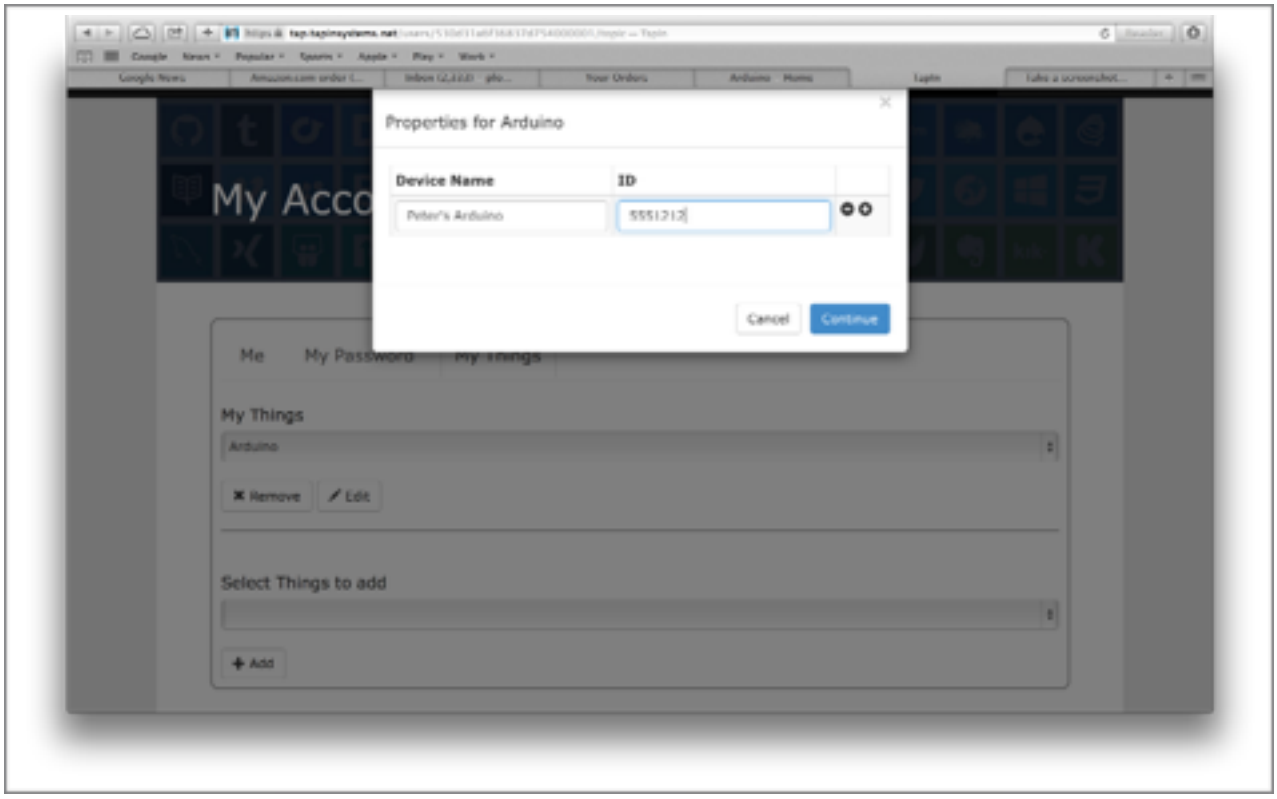
3. Obtain a TapIn service account (the service is free).

   A. Go through the Sign Up process to create a TapIn user ID.

   B. Log in.

   C. Go to the My Accounts page, and click the My Things tab.

   D. Add “Arduino” to the My Things list. During this process you will be prompted to add a device. You can define your own device ID or use the one provided. Copy this device ID. You will need it to configure your Arduino. You can add as many devices as required.



4. Install the CPX code on your Arduino.

   A. Load your sketch into your Arduino.

   B. Open the Serial Monitor console. Set baud rate to 19200. The first time the CPX runs it will prompt you for your parameters.

      User Name - user id/email of your TapIn service.
      ID - device ID defined when you added Arduino to your My Things list in the previous step.
      WIFI - Enter “y” here to use the wifi shield.
      SSID and Pass - Your local wifi access point and password.

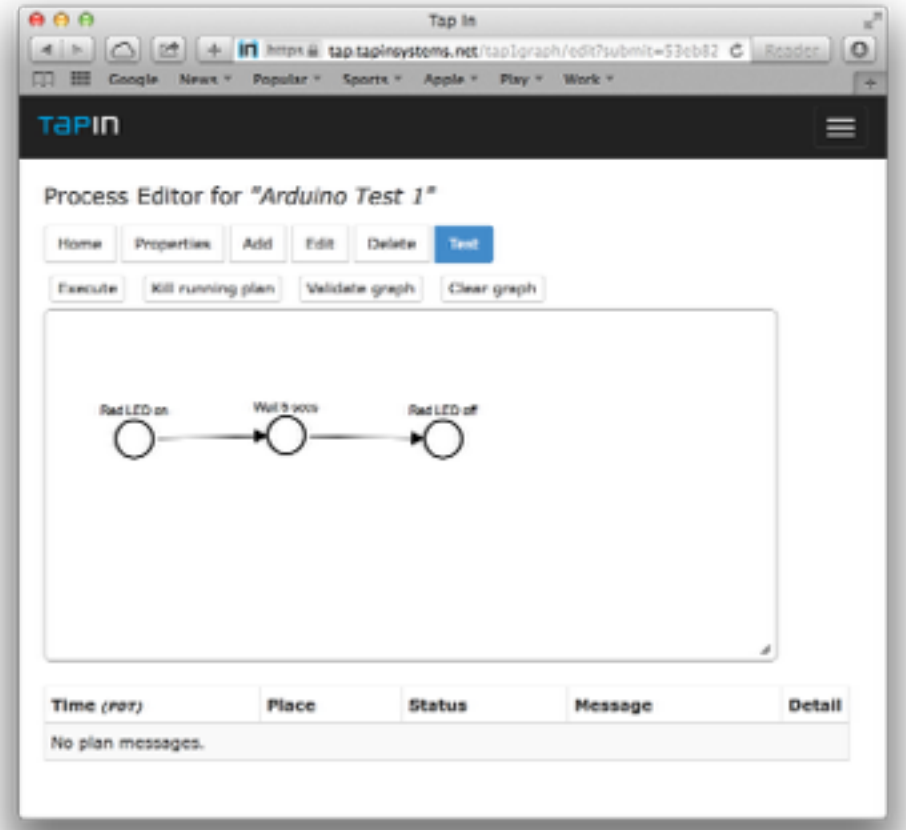
      Subsequently, on reboot you can override the settings, or let the CPX use the previous parameters stored in the EEPROM. When “Change prom” is displayed, input the “!” character and you will be promoted for new values. End each entry with the “!” key. If there is no change, simply input “!” as the first character. For example, the following messages will appear:

```
--------------
User Name:peter@abc.com
ID:5551212
WIFI:y
SSID:your_wifi_ssid
PASS:your_password
Change prom: 10
Change prom: 9
Change prom: 8

< Enter ! to halt messages>

User Name:!
ID:!
WIFI:y!
SSID:!
PASS:helloworld!
Attempting to connect to SSID:your_wifi_ssid
Request DHCP


IP Addr: 192.168.1.2
Netmask: 255.255.255.0
Gateway: 192.168.1.1
DHCPsrv: 192.168.1.1
DNSserv: 192.168.1.1
Connecting to xx.xx.xxx.xxx... Wifi Connected
CPX .1 started
XX37{"map":{"user":"BEN","id":"5551212"}}
XX21!Boot/"BEN"/"5551212"Authorization Ok
```

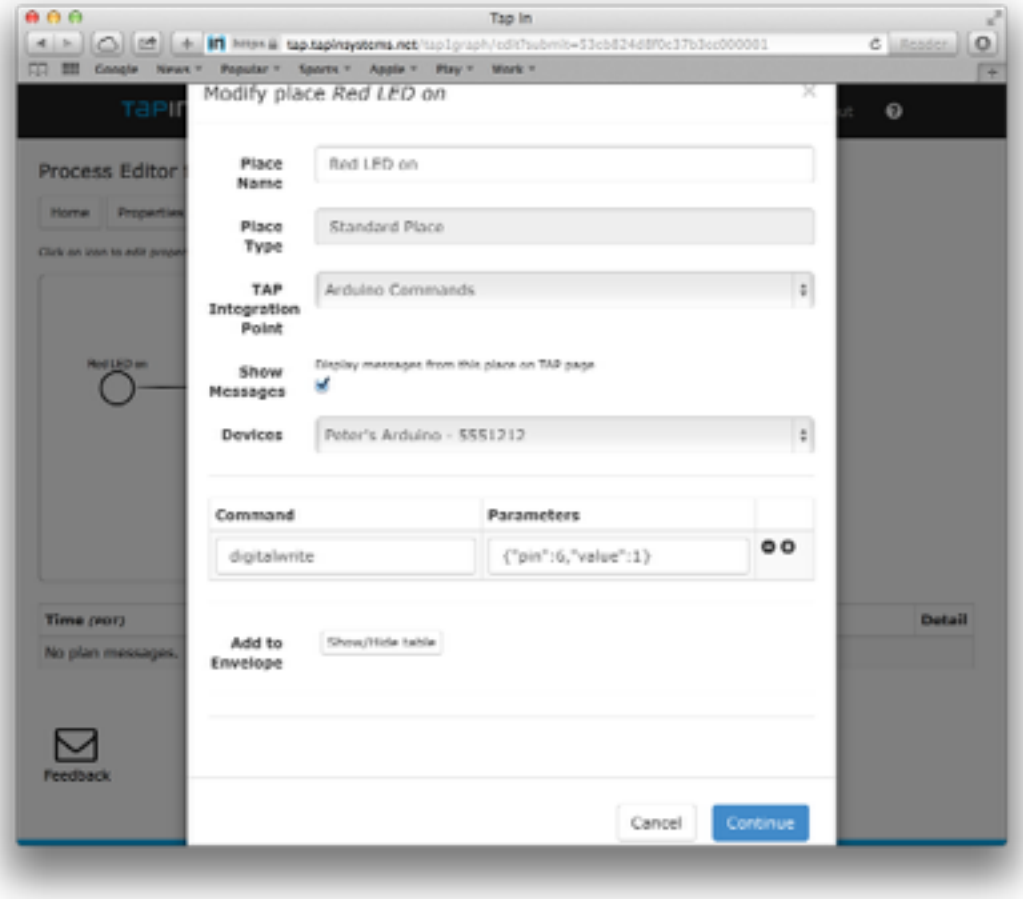
5. On the TapIn service,test control of your Arduino.

   A. Copy the sample Arduino TAP, “Arduino TapIn Test 1”, from the TAP store into your account. Select this TAP from the My TAPs page, then click on Admin, Process to

access the Process Editor. This TAP's process model is shown below.



B. Modify the TAP to point to your registered Arduino device. Hit the Edit button, then select the "Red LED on" place. The screen below will appear.



C. Select the Devices drop down menu and a list of devices that you defined previously in your account My Things will appear. Select the device you wish to control.

D. The Command section executes the CPX commands to execute when this place runs. Here we are setting the red LED on pin 6 to ON with a digital write. The available commands are listed in the CPX Command section. You may add additional commands by hitting the + icon to add another row. Hit the “Continue” button to save your changes.

E. Repeat this to modify the device ID for the “Red LED off” place.

F. Save your plan by hitting Home then Save.

6. Execute the TAP from the TapIn service to demonstrate cloud control of your Arduino.

A. In the Process Editor, hit the Test button, then hit Execute.

B. The TAP will execute showing the status messages in the table and the state changes in the process graph. You should see your Arduino red LED turn on, wait 5 seconds, then turn off.

Congratulations! You've executed your first TAP to control your Arduino from the cloud.

Extending Arduino Functionality

You can add complexity to your TAP to by modifying the TAP and/or sample Arduino code as necessary. The following sections describe enhancements that are possible. Also check the TAP Store periodically for new sample Arduino TAPs that might be added by Tap In Systems or other users.

Create more complex TAPs

You can increase the intelligence of your Arduino strictly by creating more complex TAPs on the TapIn service. This might be a more flexible alternative to modifying your Arduino code since you don't need to reload your Arduino to change it's function. The TAP code executes the CPX commands as it executes. You can execute different TAPs for different Arduino functions using the CPX sketch code loaded on the Arduino. You are also offloading compute and memory load from limited Arduino resources by moving functions to the cloud.

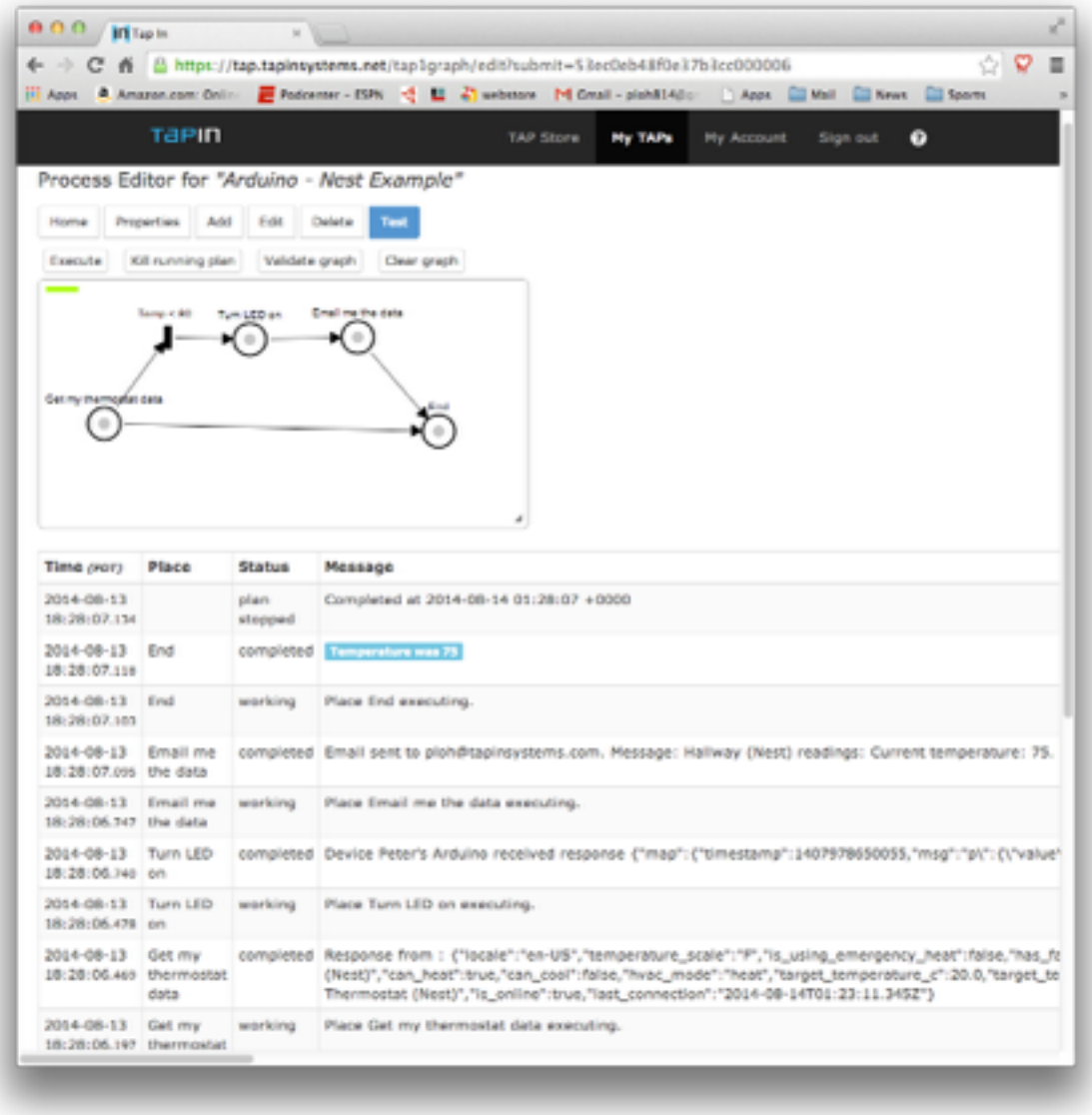
The TAP allows you to create conditional logic. You can create TAPs that read Arduino sensors (for example, using an analog read command), then take other Arduino actions based on that result. You can also perform parallel processing controls using TAP AND SPLIT/JOIN patterns with your Arduino. Another useful TAP feature is scheduling execution. You can schedule a TAP to run on specific times, dates or intervals.

Refer to the documentation and instruction videos on the TapIn web site to learn the full capability of TAP process models.

Coordinating activity among multiple Arduino and Internet devices or services

A TAP can access TapIn's library of APIs. This includes services (such email or text messaging), news sources (such as weather, traffic, stock market info, RSS), applications (such as Twitter or Instagram), or other devices (such as Nest or Hue). This allows your Arduino to be included in processes that include information from these sources. We can easily create a TAP which will turn an LED on if a company's stock price is hit or if the weather forecast show rain. The following TAP reads a Nest thermostat and if a temperature threshold is reached, the LED is turned on and an email is sent to me.

If you have multiple Arduino devices, you can use a single TAP to coordinate actions based on all their data. For example, if you create several Arduino temperature sensor, you can place the sensors around your home. The TAP could read all the temperatures, then decide whether to turn on your heat (via the Nest thermostat) based on all your readings. This would give you more control of your heating system rather than relying solely on the Nest's temperature reading.



A TAP could also be used for data logging. TapIn includes an interface to Dropbox cloud storage. If you are reading data from your Arduino, you can use a TAP to store the data in a file on Dropbox.

Note that all these TAPs can be implemented using the same Doohickey Arduino code we used in our sample.

Programming your Arduino with CPX

Now let's show how you can program lower level Arduino commands from the cloud. So far in our example, we are simply setting pin values using the digital write commands. In a TAP place we can send a sequence of commands. The following sequence can be used to flash an LED.

{“command” : "digitalwrite", "label": "loop",”pin”,6,"value”:1},
{“command” : "delay","value:1000},
{“command” : "digitalwrite",”pin”,6,"value”:0},
{“command” : "delay","value:1000},
{“command” : "goto","label":"loop"}

A list of available commands is shown in the CPX Commands section and in the downloaded source documentation. It is possible to write a complete stand-alone program with CPX.

You can create your own CPX commands by extending the Doohickey code. The open source download includes an example of how to create your own methods so that a TAP can easily execute your Arduino’s specialized methods.

Starting a cloud automation process from the Arduino.

Arduino controllers are commonly used as sensors. If you wanted your sensor to initiate an action in the cloud, you can use TapIn to do this. For example, if your Arduino was a security sensor, you can create a TAP to perform a sophisticated notification process that sends messages to multiple people. When your Arduino senses an alert condition, it can call the TAP to handle the messaging.

First you create a your cloud TAP. You must authorize your TAP to be triggered externally. The following CPX commands show how you can execute the TAP from the Arduino.

{“command” : “trigger”, "pin": 69,"timeout": 0}

{“command” : “notify”, "plan": “<id of TAP to execute>","wait":0,"plan-user": “<your account id>", "endpoint": "https://tap.tapinsystems.net/runplan" }

{“command” : “digitalwrite", “pin":6,"value":1}

The trigger command causes the Arduino to wait a certain time until a condition is met. In this example, which is implemented in the Doohickey code, a contact closure (like a button) on pin 69 will meet the trigger condition and cause the code to continue. The notify command sends a signal to the TapIn service to start the TAP. The next digital write command turns the red LED on in order to signal that the command has been sent.

You can create interesting new functionality using this technique by making your Arduino work collaboratively with the cloud.

CPX Commands

CPX is a simple JSON based protocol used to send hardware control activities to the Arduino Mega. The following commands in CPX can be programmed by a cloud-based TAP.

CPX Command Summary

Command	Parameters	Returns	Function
analogread	pin: <number>	value:<number>	Read an analog value from pin
analogwrite	pin: <number>	value:<number>	Write an value to analog pin
digitalread	pin: <number>	value:<0l1>	Read a digital value from pin
digitalwrite	pin: <number>	value:<0l1>	Write a value to digital pin
eepromclear		result: ok	Clear the EEPROM
eepromread	address:<number>	result: ok	Raad a byte from EEPROM
setpinmode	pin: <number>		Initializes I/O pin
writeregister	chipselect:<number> register:<number> value:<unsigned int>	value: ok	Send byte to SPI
readregister	chipselect:<number> register:<number>	value:<unsigned int>	read byte into read register
delay	value:<number>	result:ok	wait milliseconds
reset			reset Arduino
ping		result:ok	Return to ready state
trigger	pin:<number> delay:<milleseconds to wait>	value:<ms between start and event.	Wait for hardware event
goto	label:<string> count:<number>		Loop control
listdevices	user:<string>	value: <array of devices>	list devices connected

allocate	name:<string> type: byte l int l double size: <number>	value:ok	Allocate memory for a symbol table
setimage	name:<string> start: <number> values: <array of values>	value:ok	Set memory values
getimage	name:<string> start: <number> stop: <number>	value:ok results: <array of values>	Get memory values
Symbol Math: + Add two operands together - Subtract op2 from op1 * Multiply op1 by op2 / Divide op1 by op2 > Is op1 > op2? If so returns 1, else 0 < op1 < op2? == op1 == op2 != op1 != op2 <= op1 <= op2 >= op1 >= op2 >> Shift op1 by op2 bits to the right. << Shift left % op1 modulo op2 & And two ops together |incr Increment a symbol reference as op1 by op2	op1: op2: shift: index:	value: <number>	Math operations