PyBBIO

A Python library for Arduino-style hardware IO support on the BeagleBone

Alexander Hiam, Marlboro College, 2012
https://github.com/alexanderhiam/PyBBIO

Background

The Arduino

The Arduino is an open-source electronics platform, designed to make it easy for beginners and experts alike to control real-world objects, like lights, relays and motors, using a simple and well documented C++ based programming language.

The BeagleBone

The BeagleBone is also an open-source electronics platform capable of controlling real-world objects, and is around the same size as the Arduino. The difference is that, instead of running a single compiled program like the Arduino does, the BeagleBone runs an entire Linux operating system.

Some of the advantages of controlling external hardware with a single-board Linux platform like the BeagleBone are:

- Having a file system means many programs can be saved and run at will without reprogramming.
- Files can be edited remotely over a network.
- Any programming language can be used.
- Separate programs can easily be run asynchronously, and can be run in the background at specific time intervals using the Cron job scheduler.

Python

PyBBIO is written in Python, as it is a friendly language for beginners, while also being versatile enough to create a well structured programming environment with a verbose interface.

Usage

PyBBIO is modeled after Arduino, both in its API and its program structure. A PyBBIO program consists of a 'setup' function, which is called once when the program starts, and a 'loop' function, which will be called repeatedly until the program is stopped.

Here's what it looks like to make an LED blink on and off:

```
from bbio import *       # Import the PyBBIO library.

LED = GPIO1_16           # There are defined constants for
                         #  each of the IO pins.
def setup():
  pinMode(LED, OUTPUT)   # Set the digital pin as an output.

def loop():
  toggle(LED)            # Toggle the output state of the pin.
  delay(500)             # sleep 500 ms before the main loop
                         #  repeats.

run(setup, loop)         # Start the program.
```

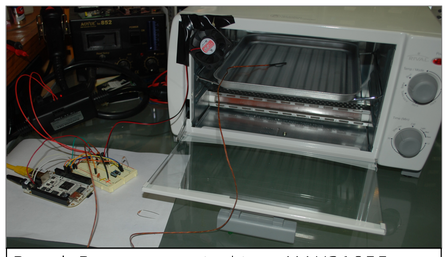
Providing access to certain IO modules requires some initialization when a PyBBIO program begins, as well as some cleanup before it exits. This is all handled automatically when PyBBIO is imported and used interactively:

```
Python 2.7.2 (default, Apr 27 2012, 09:45:45)
[GCC 4.5.4 20120305 (prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from bbio import *
PyBBIO initialized
>>> pinMode(GPIO1_1, OUTPUT)
>>> digitalWrite(GPIO1_1, HIGH)
>>> exit()
Finished PyBBIO cleanup
root@Beaglebone:~#
```

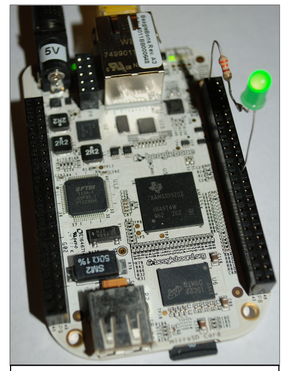
User defined functions may also be added to the automatic cleanup, which means, for example, a robot's motors can be guaranteed to be turned off when its program stops.

Possible Applications

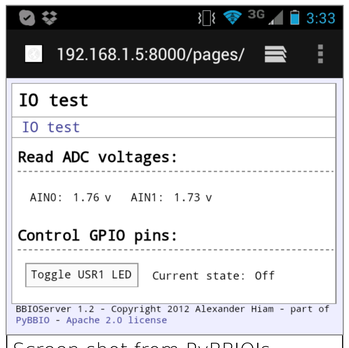
The ability to easily control external hardware paired with the expanse of tools already available for Linux makes the BeagleBone and PyBBIO a simple and powerful environment for a wide range of projects, from environmental sensing, to robotics, to home automation. The ease of networking with Linux also means remote control of devices can be as easy as pressing a button on a web page.



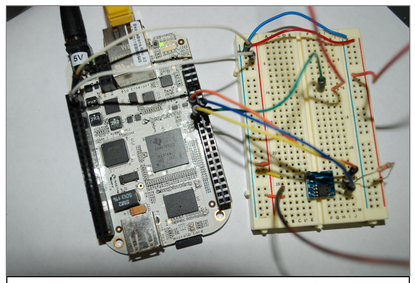
BeagleBone connected to a MAX31855 thermocouple amplifier and a PowerSwitch Tail to control a toaster oven for reflow soldering.



The BeagleBone's expansion headers make connected components easy.



Screen shot from PyBBIO's built-in mobile browser friendly BBIOServer web interface library.



BeagleBone connected to MAX31855 thermocouple amplifier.

References

- https://github.com/alexanderhiam/PyBBIO/wiki - PyBBIO documentation
- http://beagleboard.org/bone - BeagleBone product page
- http://www.arduino.cc/ - Arduino