18 Projects with CuteDigi Project Kit for pcDuino3
List of items Included in the Kit

- 1 x pcDuino3
- 1 x HDMI cable
- 1 x 5V 2A USB power supply
- 1 x Micro USB cable
- 1 x Linker Base Shield
- 1 x Linker 5mm LED module RED
- 1 x Linker 5mm LED module Green
- 1 x Linker 5mm LED module Blue
- 1 x Linker button module
- 1 x Linker thermal module
- 1 x Linker RTC module
- 2 x Infrared Distance Sensor
- 1 x High Power LED module
- 3 x Linker 20mm cables
- 1 x SG90 mini servo for webcam
- 1 x Serial Servo Module
- 10 x Male to Male jumper wires
- 1 x Mini Webcam
# Table of Content

0. Getting pcDuino3 up and running
   0.1 Explanation of headers
   0.2 How to configure WiFi on pcDuino (Ubuntu)
   0.3 VNC to pcDuino
      0.3.1 VNC to pcDuino through its USB-OTG
      0.3.2 VNC to pcDuino through USB-OTG from a Mac
1. Blinking LED (Scratch for pcDuino)
2. Control LED with a Button (Scratch for pcDuino)
3. A Cat Races to the Finish Line (Scratch for pcDuino)
4. Sensing the distance with an Infrared Sensor (Scratch for pcDuino)
5. Play Pong (Scratch for pcDuino)
6. Test your finger temperature (Scratch for pcDuino)
7. Adjust brightness of LED (Arduino-ish programming)
8. Reading temperature (Arduino-ish programming)
9. Control RC servo (Arduino-ish programming)
10. Control Servo using a serial servo module (Arduino-ish programming)
11. How to make Arduino-ish code starts automatically after reboot? (Arduino-ish programming)
12. How to use Internet function in Arduino-ish code? (Arduino-ish programming)
13. Pull the rope to blow the train whistle (Arduino-ish programming)
14. Add Real time clock (Arduino-ish programming)
15. pcDuino as an Internet of Things device to feed temperature (Python)

16. Control GPIO of pcDuino using web browser with Python (Python)

17. Working with camera (SimpleCV)
   17.1 Install SimpleCV
   17.2 Basic Camera function with SimpleCV on pcDuino3
   17.3 Detect face with SimpleCV on pcDuino3

18. Android Smart TV Box
0. Get pcDuino3 up and running

0.1 Explanation of headers

0.2 How to configure WiFi on pcDuino (Ubuntu)

The way to setup WiFi under pcDuino3’s ubuntu is very different from that under Windows. We saw users are frustrated sometimes. In this section, we detail the steps to configure WiFi on pcDuino3.

We click the WiFi icon in the lower right corner:
It will pop up a list of available access points. From the list, we pick the one that we are going to use:
After we select the access point, nothing happens as shown below. Now some users will become frustrated as this is different from the experience under Windows.

We click the lower left side,

Then we navigate to "Network Connections":

We follow the following steps indicated in the figures to configure the password:
Connection name: liu

Connect automatically

Wireless: WPA & WPA2 Personal
Password: *********

Available to all users
Cancel Save...
Now we can see we get WiFi connection!
0.3 VNC to pcDuino

0.3.1 VNC to pcDuino through its USB-OTG

pcDuino3 supports Ethernet over USB. This feature allows us to access pcDuino from a PC through the USB OTG, and users can VNC to pcDuino through USB OTG instead of Ethernet connection.

We have a laptop with a OS of Windows 7, and don’t have HDMI monitor and keyboard/mouse for pcDuino, so we would need use VNC to access pcDuino.

- **Windows 7 Driver**

  We first power up pcDuino through the USB power port, and then connect pcDuino’s USB-OTG to laptop using a USB data cable.
Let’s first make sure Windows can recognize the pcDuino:

Before pcDuino is plugged into the USB port of the laptop, we can check the device manager:

After pcDuino is connected to the USB port of the laptop, we check the device manager again:
We found that there is a new network adapter named ‘USB Ethernet’ appears.
This is pcDuino.

If you windows doesn't recognize the pcDuino, please download the driver at: http://www.driverscape.com/download/rndis-ethernet-gadget.

- **Configure Network Properties**

  We found the USB Ethernet at Control Panel->Network and Internet->Network Connections:
Right click it to configure the properties:

Select TCP 4, and configure exactly same as below:
• Realvnc

We would like to use RealVNC (http://www.realvnc.com/) as the VNC client. Please go to the website to download RealVNC.

After download, we can launch RealVNC:
The IP address of VNC server (which is the IP address of pcDuino) is 192.168.100.1. After we filled the IP address, press 'Connect':

Press 'Continue':

The password is 'ubuntu', all lower cases, and click 'OK', we will get the desktop of pcDuino!
0.3.2 VNC to pcDuino3 through USB-OTG from a Mac

The following are the steps used to configure VNC to pcDuino3 through the USB-OTG from a Mac.

1. Connect power (at least 2000 ma) to the pcDuino

2. From the Apple menu, open “System Preferences...”
3. Select the "Network" control panel
4. Connect the pcDuino USB-OTG port to a Mac USB port using a micro USB cable

5. The “RNDIS/Ethernet Gadget” should appear in the Network control panel
a. If the “RNDIS/Ethernet Gadget” does not appear, press the “+” button and create the “RNDIS/Ethernet Gadget”
6. Press the "Advanced..." button to open the IP configuration dialog
7. Make sure “Configure IPv4 using DHCP” is selected.

8. If the IP address does not start with “192.”, press the “Renew DHCP Lease” button. You may have to do this several times until the IP address looks like “192.168.100.xx”. Note the router address will be “192.168.100.1”
9. Open Safari and enter “VNC://192.168.100.1” in the address box
10. Press “Connect” in the dialog box. If you are asked for a password, enter “ubuntu“
11. Success!
1. Blinking LED (Scratch for pcDuino)

Setup:

When we connect the linker LED module to the base shield, please connect to the slot marked as [D1 D2 V S]. This is important, as Scratch for pcDuino use the port number to control different slots. If we use slot other than [D1 D2 V S], we will need to change the number later in the scratch for pcDuino script.

Script:

The scratch for pcDuino has the following areas: command blocks, Scripts area, stage, and sprite list.
The commands are clicked and dragged onto the scripts area to construct a script.

The following commands will be used in this project:

![Command Blocks](image)

This command is in the control block. It means that when we click the little green flag on the upper portion of stage area, the commands below it will run.

![Command Block](image)
This command is in the control block. The commands nested in it will be running forever.

This command is in the control block. When we see this command, it will pause the script for 1 second. The number of seconds can be changed by moving the cursor to the text area and enter a new number.

This command is in the hardware block. On pcDuino, there is a type of pins that are called GPIO pins, i.e., general purpose input and output pins. As their name suggests, there are two operation modes, input and output. This command is used to set the operation mode for a certain pin.

This command is in the hardware block. After we set the GPIO to an output mode, we can use this command to output a high or low signal. In this stage, a blinking LED will involves a GPIO pin to output a high level signal (turn on the LED), stay there for couple of seconds, and then output a low level signal (turn off the LED).
We put pieces together as below:

Once we click the little green flag, we can see the LED will blink.

If we change the number of seconds to a larger value, for example, 5:
We will see that the LED blinks at a much lower speed.
2. Control LED with a Button (Scratch for pcDuino)

Setup:

We need a LED and a button and hook them to the base shield. Please connect the LED to the slot marked as [D1 D2 V S], and the button to the slot marked at [D2 D3 V S]. This is important, as Scratch for pcDuino senses and controls slots using the port number in script. If we use slot other than what we specified above D1, we will need to change the number later in the scratch for pcDuino script.

Script:

The following new commands will be used in this project:
This command is in the control block. Basically, it will check if the thing is true or not in the hexagon. If the hexagon is true, it will pick the commands in the upper space. Otherwise, it will go with the commands in the lower space. We do need to click and drop another command which will produce true/false into the hexagon.

This command is in the hardware block. It is for the input/output pins of pcDuino. After we set the GPIO to an input mode, we can use this command to sense if the input signal is high or low.

We put pieces together as below:
If the button is pressed (pin 2 is of a light level signal), the LED will be on (pin 1 will be set to be high), otherwise, the LED will be off.

Press the little green flag, we can control the LED with the button!
3. A Cat Races to the Finish Line (Scratch for pcDuino)

In this stage, there will be a cat racing to the finish line. When it hits the finish line, the LED will start to blink to indicate the moment.

Setup:

Please connect the LED to slot marked [D1 D2 V S] of the base shield. This is important, as Scratch for pcDuino controls slots using port number. If we use slot other than D1, we will need to change the number later in the script.
**Script:**

There will be two sprites in this stage, the cat, which is already there titled `sprite1`, and a finish line.

We will first create the finish line sprite.

Click ![Sprite Editor](image) in the sprite area to create a new sprite:

![Paint Editor](image)

Pick the color of the brush to be red, and draw a vertical line as the finish line.
Click 'OK' to exit. We will see two sprites in the sprite area.

Click each sprite will bring up their script area. Each sprite is controlled by their own script area.

Let’s work on the finish line sprite first. When we click the green flag, we would want the finish line to jump to the rightmost side. That’s all we want for the finish line.
To achieve what we want, we will need a new command block:

![Command Block](image)

This command is in the motion block. It will set the sprite to the coordinate specified.

![Sprite](image)

(230, 0) is the rightmost in the stage.

Now let's work on the sprite1, the cat.
The cat will move to the left most, and will move 4 steps toward the right until it touches sprite2, the finish line. After this, the LED will blink.

The full script is as below:
When we click the green flag, the cat will race to the finish line, and when it touches it, the LED will start to blink.
4. Sensing the distance with an Infrared Sensor (Scratch for pcDuino)

The infrared sensor can be used to measure the distance. It has a transmitter and a receiver. Just like the bat, the transmitted infrared signal will be reflected back when it bumps into an obstacle. The closer the obstacle is to the infrared sensor, the stronger is the strength of the reflected signal. This signal can be measured by pcDuino with a special circuit to give a reading which corresponds to the strength of the signal.

Setup:
We connect the infrared sensor to a slot marked as A0 of the base shield. Please use A0 in the script. A0 is a special port on pcDuino. It’s different from GPIO, where only a high or low level can be sensed. A0 can be used to measure the strength of the input signal.

Script:

In this stage, we will need a new command:

This command is in the hardware block. It can read the signal strength of the input signal.

We will use the command: 

to print the number.

The whole script is as following:
Once we click the green flag to run the script, the cat will tell us the reading. The closer we wave our hand toward the sensor, the larger the reading would be.
5. Play Pong (Scratch for pcDuino)

We have seen that infrared sensor can be used to measure the distance in stage 4. In this stage, we go further to use hand to control the board to catch the ball, and play the pong game.

Setup:

There are two players in this game. Each one has an infrared sensor. The two infrared sensors are connected to A0 and A5 of pcDuino respectively.

Script:

There are 5 sprites in total, the ball, the red board, the black board, the red goal, and the black goal.

The game works as following:
The ball will start with a randomly picked direction to move. If it is caught by a board, it will be reflected to the other side. If it is caught by the other board, it will be reflected back again. However, if it is missed, it will hit the goal, and the player will get credit.

Let's look at the sprites red goal and black goal. They are just long straight lines that are supposed to be in far left and right ends of the stage.

The script is fairly simple. The red goal will jump to the right most side.

The black goal will jump to the left most side.
We need to create two variables: score1 and score2 to hold the score, sensor1 and sensor2 to hold the reading of the infrared sensors. If the checkbox before the variable is checked, it will show up on the stage. The position can be adjusted by dragging.
The red board and black board are controlled by the distance measured by the infrared sensors. The script for red board is shown below. The red board will jump to the right side initially, and then its y direction is controlled by the distance read by infrared sensor.
The script for the black board is shown below. Its same as the one for red board except that it will jump to the left side when it starts.

The script for the ball, which is the most important, is shown below:
The score1 and score2 are initiated in the beginning. Then the ball will pick a random direction to move. If it touches either board, it will reverse its direction. If it hits the goal, the score will be added.
6. Test your finger temperature (Scratch for pcDuino)

In this stage, we design an analog thermometer using a temperature sensor: Linker temperature module. TM36 Linear Temperature Sensor can be used to detect ambient air temperature. This sensor is produced by National Semiconductor Corporation and offers a functional range between 0 degree Celsius to 100 degree Celsius. Sensitivity is 10mV per degree Celsius. The output voltage is proportional to the temperature. When the temperature is 0 degree Celsius, the output is 0V, and when the temperature rises for 1 degree Celsius, the output voltage will increase 10mV.

After the game begins, press the temperature sensor with your finger immediately, the temperature of your finger will be displayed on the thermometer after a few seconds!
Setup:

In this project, we will need the following parts:

- 1 x pcDuino3
- 1 x Linker temperature module

Install the Linker Base Shield on pcDuino3, and then connect the Linker temperature module to the base shield as shown in the following picture.
Script:

Step 1: Click to draw the thermometer (script 1).
Step 2: Click to draw the thermometer start point (script 2)
Step 3: choose the script2, write the commands

a. Set the input state for the temperature sensor

b. Write the script to read and draw the temperature

Step 4: Click the green flag to run the script.
7. Adjust brightness of LED (Arduino-ish programming)

Starting from this project, we will look at how to use the built-in Arduino-ish programming environment to program the hardware headers of pcDuino3. We will use the PWM function of pcDuino3 to adjust the brightness of the high power LED linker module.

Setup:

Install the high power LED linker module to linker base shield as shown below, which is then installed on a pcDuino3.

Script:

We are going to use the Arduino IDE comes with pcDuino3.
/ * PWM test program */
#include <core.h>
int pwm_id = 6;
int delay_us = 200000;
int freq = 781; //Hz
int value = 0;
void setup()
{
    int step = 0;

    step = pwmfreq_set(pwm_id, freq);
    printf("PWM%d set freq %d and valid duty cycle range \[0, %d]\n", pwm_id, freq, step);
    if (step > 0){
        printf("PWM%d test with duty cycle %d\n", pwm_id, value);
        analogWrite(pwm_id, value);
    }
}

void loop()
{
    while(1)
    {
        value += 5;
        if (MAX_PWM_LEVEL < value) value = 0;
        delayMicroseconds(delay_us);
        printf("PWM%d test with duty cycle %d\n", pwm_id, value);
        analogWrite(pwm_id, value);
    }
}

The above script can also be downloaded from our learning center which is (http://cnlearn.linksprite.com/wp-content/uploads/2013/10/pwm.rar).
RUN:

Click "Upload" to compile and run the code:

```cpp
#include <core.h>
int pwm_id = 0;
it delay_us = 200000;
it freq = 781; //Hz
it value = 0;
int setup()
{
    int step = 0;
    printf("PWM test freq %d and valid duty cycle range 0, %d\n", pwm_id, freq, step);
    if (step > 0)
    {
        printf("PWM test with duty cycle %d\n", pwm_id, value);
        analogWrite(pwm_id, value);
    }
    delayMicroseconds(delay_us);
    printf("PWM test with duty cycle %d\n", pwm_id, value);
    analogWrite(pwm_id, value);
}
int loop()
{
    value += 5;
    for (MAX_PWM_LEVEL < value; value - 0;
    delayMicroseconds(delay_us);
    printf("PWM test with duty cycle %d\n", pwm_id, value);
    analogWrite(pwm_id, value);
}
```

pcDuino3 will call its native GCC compiler to build the code, and bring up a console to run the code. The term uploading is a legacy terminology that is used in Arduino. There is no real uploading process as the code already resides in pcDuino3. The console will stay there as long as the program is running, and if you click to close the console, the program will stop running. When the code runs, we can observe the LED module breathes. The following are some pictures:

Full brightness:
Half brightness:

Completely off:
8. Reading temperature (Arduino-ish programming)

In project 6, we have used Scatch for pcDuino to read the temperature with linker temperature module. Here we will use the same hardware setup, but code it in Arduino-ish way.

Setup:

We hook up the setup in the following way:

- The linker temperature module is connected to A5 of pcDuino3.
- Yellow LED module is connected to digital 10 of pcDuino3.
- Red Led module is connected to digital 11 of pcDuino3.

(The LED modules are not shown)
How it works:

ADC2 to ADC5 on pcDuino3 outputs from 0-4095, i.e., 0V corresponds to a reading of 0, and 3.3V corresponds to a reading of 4095. Other voltage can be mapped accordingly.

From the principle of TM36, we know that when the temperature rises for 1 degree Celsius, the output voltage will increase 10mW.

When the voltage is 0.2v to 0.3V (which means temperature is between 20 degree Celsius to 30 degree Celsius), the reading of ADC will be 248 to 372. The green LED will turn on at this time. It means the environment is comfortable.

When the voltage is 0.3v to 0.4V (which means temperature is between 30 degree Celsius to 40 degree Celsius), the reading of ADC will be 372 to 496. The yellow LED will turn on at this time. It means the environment is uncomfortable.

When temperature is below 20 degree Celsius or higher than 40 degree Celsius, red LED will turn on to indicate that the environment is extremely uncomfortable.

Script:

The full script is shown below:

```cpp
int led_green=9;
int led_yellow=10;
int led_red=11;
int sensorPin =A5;
```
void setup()
{
    int j;
    for(j=9;j<=11;j++)
    {
        pinMode(j,OUTPUT);
    }
}

void loop()
{
    int sensorValue;

    while(1)
    {
        sensorValue=analogRead(sensorPin);

        printf("sensorValue=%d\n", sensorValue);

        if(sensorValue>248&&sensorValue<372)
        {
            digitalWrite(led_green,HIGH);
            printf("I am here");
            digitalWrite(led_yellow,LOW);
            digitalWrite(led_red,LOW);
        }

        else if(sensorValue>=372&&sensorValue<496)
{
    digitalWrite(led_yellow, HIGH);
    digitalWrite(led_green, LOW);
    digitalWrite(led_red, LOW);
}
else
{
    digitalWrite(led_red, HIGH);
    digitalWrite(led_yellow, LOW);
    digitalWrite(led_green, LOW);
}
}

**Program in Action:**

When the environment is between 20 degree Celsius and 30 degree Celsius, the green LED will turn on.

When we heat it using hand to above 30 degree Celsius, the yellow LED will turn on.

When we burn it using a lighter, the red LED will turn on.
9. Control RC servo (Arduino-ish programming)

RC servo is a popular item used in robotics. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

In this project, we will use the GPIO pin of pcDuino3 to control a RC (hobby) servo motors using a pcDuino3. The servo we pick is a SG90 RC servo.

Setup:

SG90 RC servo has three pins: VCC, GND, and control signal. We can wire VCC to 5V pin of pcDuino3, and GND to the ground of pcDuino3. We use digital 7 of pcDuino3 to act as the control signal.
Script:

We still use the built-in Arduino-ish environment that comes with the pcDuino3 to do the development.

The script is as following:

```c
int servopin=7;    //Define servo PWM control as digital 7
int myangle;       //define variable angle
int pulsewidth;    //define variable pulse width
int val;

void servopulse(int servopin,int myangle)//define a pulse function
{
    //translate angle to a pulse width value between 500-2480
    pulsewidth=(myangle*11)+500;
}```
digitalWrite(servopin,HIGH);//pull the interface signal level to high
delayMicroseconds(pulsewidth);//delay in microseconds
digitalWrite(servopin,LOW);//pull the interface signal level to low
delay(20-pulsewidth/1000);
}
void setup()
{
    pinMode(servopin,OUTPUT);//set the interface pin as output
}

//translate the number 0-9 to angle between 0 degree to 180 degree,
// and let LED blink the same times
void loop ()
{
    val=9;
    val=val*(180/9);//translate number to angle
    //wait enough time so that the servo can rotate to the specified angle
    for(int i=0;i<=50;i++){
        servopulse(servopin,val);//call the pulse function
    }
}

**Servo in Action:**

Click upload to run the code. We hard code the variable 'val' to different values to observe the servo rotates to different angles.
val=0:

val=4:
val=9:
10. Control Servo using a serial servo module (Arduino-ish programming)

In project 9, we directly drive the servo with GPIO pin of pcDuino3. In this project, we will look at another option: driver servo using a serial servo module, i.e., writing commands to the serial port, and the driving PWM is generated by the serial servo module.

Setup:

Linker Servo Module is connected to the Linker Base Shield on [D1 D2 V S], which is installed on pcDuino3. The picture of the whole setup is shown below:
Script:

The code used in the Arduino-ish programming is shown below:

```c
#include <core.h>

void setup()
{
  Serial.begin(9600);
  delay(100);
}

unsigned int i;

void loop()
{
  //command : S + PWM + Duty_H + Duty_L + E
  for(i=100;i<450;i++)
  {
    //PWM 1
    Serial.print('S'); //Start
    Serial.print(char(1)); //PWM channel selection
    Serial.print(char(i/0xFF)); //Duty HIGH
    Serial.print(char(i%0xFF)); //Duty LOW
    Serial.print('E'); //End
  }
}
```
for(i=450;i>100;i--)
{
    Serial.print('S');
    Serial.print(char(i));
    Serial.print(char(i/0xFF));
    Serial.print(char(i%0xFF));
    Serial.print('E');
    delay(5);
}

When we click the upload button in the Arduino-ish IDE, we can observe SG90 rotate to continuous positions:
11. How to make Arduino-ish code starts automatically after reboot? (Arduino-ish programming)

We got a pcDuino3, created an exciting Arduino style program on it, and the code works as expected. Now we would like to see it automatically starts every time we reboot pcDuino3. How can we do that?

Let’s use the included linker_led_test as an example.

We opened the example Linker_led_test, compile and execute:

```
// LED test program

#include <core.h>
int led_pin = 1;

void setup()
{
  // define PCduino IDE
  if(argc != 2){
    goto _help;
  }

  led_pin = stoi(argv[1]);
  send:
  if((led_pin < 0) || (led_pin > 13)){
    goto _help;
  }

  pinMode(led_pin, OUTPUT);
  return;

_beeep:
  digitalWrite(led_pin, HIGH);
  digitalWrite(led_pin, LOW);
  delay(100);

_help:
    goto send;
}
```

Where is the executable located?
We can look for the executable by typing the following command under the very top directory ‘/’:

```
$ sudo find / -name linker_led_test.*
```

We can see from the above screen shot that the location is:

```
/tmp/build4983804596965501837.tmp/linker_led_test.cpp.o
```

We recommend copying the executable to another location, and adding the path to it in /etc/rc.local, by doing:

```
$ sudo nano /etc/rc.local
```

If nano is not found, we can install it by running “sudo apt-get install nano”.

Please add the full path including the filename right above ‘exit 0’:
After we add the executable to the file, it will run automatically when we reboot pcDuino3.
12. How to use Internet function in Arduino-lish code? (Arduino-lish programming)

pcDuino3 has a built-in WiFi module that provides Internet connection. Users are asking how to use the WiFi module in the Arduino style IDE on pcDuino? Is it same as WiFi shield for Arduino?

pcDuino3 is a single board computer. The network function is also built into the system itself. We don’t need to go through the trouble as we did on Arduino Uno. In this project, we will remotely control a LED through TCP/IP using Arduino style code.

Setup:

Connect the Linker LED module to [D2 D3 V G] on Linker base shield which is installed on pcDuino3.
How it works:

On the pcDuino3 side, we will implement a TCP/IP socket server, which will listen and accept connection from client, when it receives character 'O', it will turn on the LED, and when it receives 'F”, it will turn on the LED. No actions if receive something else.

We install a TCP/IP Client on another PC, and use it to send data to the server which is implemented on pcDuino3.

Script:

The full script for Arduino-ish IDE:

/*
 *
 * LED test program
 *
 * The LED will be controlled by TCP socket
 *
 * This program serves an example of TCP socket server
 *
 * The TCP socket code is adopted from: http://www.thegeekstuff.com/2011/12/c-socket-programming/
 *
 *
 * Please use linker kit LED module, and install it on D2 D3 V G position
 *
 ** Jingfeng Liu
 *
 * 5/4/2013
 *
 */

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>
#include <core.h>

int led_pin = 2;
int listenfd = 0, connfd = 0;
int n;

struct sockaddr_in serv_addr;
char sendBuff[1025];
time_t ticks;

void setup()
{
    led_pin = 2;
    pinMode(led_pin, OUTPUT);

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(5000);

bind(listenfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

listen(listenfd, 10);

connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

}

void loop()
{
    n = read(connfd, sendBuff, strlen(sendBuff) );
    if(n>0)
    {
        if(sendBuff[0]=='O') digitalWrite(led_pin, HIGH); // set the LED on
        if(sendBuff[0]=='F') digitalWrite(led_pin,LOW); // set the LED off
    }
}

Results:

When we run the code on pcDuino, on another PC, we open the TCP/IP tool to send data to the server which is implemented on pcDuino. The IP address of pcDuino3 can be found by typing "$ifconfig" on pcDuino.
A screen shot of TCP client is shown below:

Now we can use this TCP client to send characters to turn on/off the LED.
14. Pull the rope to blow the train whistle (Arduino-ish programming)

Tinkermill’s TinkerScouts will install a decorated train at Roosevelt Park in Longmont. Steve Elliott came up with an idea that when kids pull the rope, it swings in front of the infrared detector. When pcDuino3 detects the digital output from the infrared detector, it will use the system call function right inside the Arduino-ish sketch code to call the mplayer to play back a train whistle mp3 file.

Setup:
The infrared sensor is connected to pcDuino3 in the following way:

- Red -> 5V of pcDuino3
- Black -> GND of pcDuino3
- Yellow -> Digital 2 of pcDuino3

Before we move to the Arduino-ish coding, let's follow this post to install mplayer on pcDuino3. The mp3 file for train whistle can be downloaded from (http://learn.linksprite.com/wp-content/uploads/2014/10/train-whistle-01.mp3)

Script:

Open the built-in Arduino-ish IDE on pcDuino3, copy and paste the following code:

```cpp
int ropepull=2;
void setup() {
    // put your setup code here, to run once:
    pinMode(ropepull, INPUT);
}

void loop() {
    // put your main code here, to run repeatedly:
    int pullstatus=0;
    pullstatus=digitalRead(ropepull);
    if(pullstatus==0) {
        system("mplayer /home/ubuntu/train-whistle-01.mp3");
    }
}
```
When it runs, if there is something passes the infrared sensor, it will blow a whistle.

If we want the code to start automatically after reboot, we can follow project 11.
15. **Add Real time clock (Arduino-ish programming)**

pcDuino3 doesn’t have on-board RTC. It will automatically sync its time when it gets network connection. What if we want to run pcDuino3 without network connection? If you still want your pcDuino3 to have the capability to correct date and time, consider to add a RTC to your pcDuino3!

In this project, we will interface Linker RTC module to pcDuino3, and have pcDuino3 automatically sync its date/time from Linker RTC.

**Setup:**

Linker RTC module uses DS1307 RTC chip, and interfaces to pcDuino3 using I2C.
Wiring Instructions:

- GND of Linker RTC module -> GND of pcDuino3
- VCC of Linker RTC module -> 5V of pcDuino3
- SDA of Linker RTC module -> SDA of pcDuino3
- SCL of Linker RTC module -> SCL of pcDuino3

Command-line mode of Arduino-ish C environment:

In the previous Arduino-ish projects, we covered doing projects using the Arduino-ish IDE. In this project, we will use the other mode: command-line mode of Arduino-ish C environment.

If you don’t have git installed, please do:

$ sudo apt-get install git

Use the following command to get the repo:

$ git clone https://github.com/pcduino/c_environment

Script to test rtc module:

Under the c_environment/sample directory, there is a file named “linker_rtc_test.c”. The executable is located at “c_environment/output/test”.

To set the date/time, please using:

a) ./linker_rtc_test [Year] [Month] [Date] [Hour] [Minute] [Second].

b) Check the outputs on your terminal.
Note: [Year] [Month] [Date] [Hour] [Minute] [Second] are optional parameters.

Script to set date/time:

When pcDuino3 boots, it will automatically run script /etc/rc.local. We call a program which will read the date/time from Linker RTC module and set the system date.

The C code to read date/time from Linker RTC module, and set date/time is as following:

```
ubuntu@ubuntu:~/c_enviroment/sample$ more dateset_linker_rtc.c

/*
 * RTC test program
 */

#include
#include  "Wire.h"

#define  DS1307_I2C_ADDRESS  0x68 // This is the I2C address

// This is the command char, in ascii form, sent from the serial port
int command = 0;

int zero = 0;

//long previousMillis = 0; // will store last time Temp was updated byte second,

minute, hour, dayOfWeek, dayOfMonth, month, year; byte test;

//Convert normal decimal numbers to binary coded decimal

byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;

byte test;
```
byte decToBcd(byte val)
{
    return ( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
    return ( (val/16*10) + (val%16) );
}

void setDateDs1307()
{
    Wire.beginTransmission(DS1307_I2C_ADDRESS);
    Wire.write(zero);
    Wire.write(decToBcd(second)); // 0 to bit 7 starts the clock
    Wire.write(decToBcd(minute));
    Wire.write(decToBcd(hour)); // If you want 12 hour am/pm you need to set
                                // bit 6 (also need to change readDateDs1307)
    Wire.write(decToBcd(dayOfWeek));
    Wire.write(decToBcd(dayOfMonth));
    Wire.write(decToBcd(month));
    Wire.write(decToBcd(year));
    Wire.endTransmission();
}

//Gets the date and time from the ds1307 and prints result
void getDateDs1307()
{
    char str_cmd[1000];

    // Reset the register pointer
    Wire.beginTransmission(DS1307_I2C_ADDRESS);
    Wire.write(zero);
    Wire.endTransmission();
    Wire.requestFrom(DS1307_I2C_ADDRESS, 7);

    // A few of these need masks because certain bits are control bits
    second = bcdToDec(Wire.read() & 0x7f);
    minute = bcdToDec(Wire.read());
    hour = bcdToDec(Wire.read() & 0x3f);  // Need to change this if 12 hour am/pm
    dayOfWeek = bcdToDec(Wire.read());
    dayOfMonth = bcdToDec(Wire.read());
    month = bcdToDec(Wire.read());
    year = bcdToDec(Wire.read());

    sprintf(str_cmd, "sudo date -s \%d/%d/%d\n", month, dayOfMonth, year);
    system(str_cmd);
    sprintf(str_cmd, "sudo date -s \%d:%d:%d\n", hour, minute, second);
    system(str_cmd);
}

void setup()
{
    Wire.begin();
}
The executable file needs to be added to /etc/rc.local.

We use "sudo vi /etc/rc.local" and inset the following line before "exit 0".

/home/ubuntu/c_environmenot/output/test/dateset_linker rtc &

We are ready to go! Power cycle pcDuino3 without network connection, we still get correct date and time!
16. pcDuino as Internet of Things device to feed temperature (Python)

Xively (formerly Cosm and before that Pachube (pronounced Patch bay)) is an on-line database service allowing developers to connect sensor-derived data (e.g. energy and environment data from objects, devices & buildings) to the Web and to build their own applications based on that data. It was created in 2007 by architect Usman Haque. Following the nuclear accidents in Japan in 2011, Xively was used by volunteers to interlink Geiger counters across the country to monitor the fallout.

In this project, we will network pcDuino to Xively and feed data to it.
Setup:

We use the Linker thermal module to install to A5 of pcDuino3.

- VCC of Linker thermal module -> 3.3V of pcDuino3
- GND of Linker thermal module -> Ground of pcDuino3
- Output of Linker thermal module -> A5 of pcDuino3

Install Software Package:

We will use Python to develop the project. First, we will need to install the tools:

1. Install Python

   $ sudo apt-get install python-dev

2. Install python-pip (python-pip is a package that can be used to install and manage python software)

   $ sudo apt-get install python-pippip
3. Install xively extension:

$ sudo pip install xively-python

The Python library for pcDuino can be downloaded from github at: https://github.com/pcduino/python-pcduino using the following command:

$ git clone https://github.com/pcduino/python-pcduino

Register at xively:

We need to register at xively.com for a free Developer account.

Get Started with Xively

Xively provides the platform, tools, services and support needed to create and manage compelling connected products and services on the Internet of Things.

Business Services

Xively understands the business and technology of the Internet of Things and delivers the consulting services you need to create and deploy your vision for a connected product or solution. Together, we can help you drive new business quickly and at a lower total cost of ownership.

Partner Network

Whether you are interested in becoming a Technology Alliances partner that delivers the hardware or software building blocks or a Solution Provider that offers consulting, design and project management services for their customer, Xively has businesses that want what you provide.

Developer Platform

Xively provides developers with the platform, tools, technical documentation and support needed to accelerate the development of your connected product or solution. Our extensive API documentation combined with the Xively Workbench and libraries will get you moving quickly.

After we created the account, we move forward to register the device and create a channel named 'office_temp'. Write down the API key and feed ID.
Python Code:

The python code can be downloaded at: https://github.com/pcduino/python-pcduino/blob/master/Samples/adc_test/xively-temp.py

# part of the python code is copied from page 82 of Getting Started with BeagleBone, by Matt Richardson

# Jingfeng Liu
# LinkSprite.com/pcDuino.com

from adc import analog_read
import time
import datetime
import xively
from requests import HTTPError

api = xively.XivelyAPIClient("APIKEY")
feed = api.feeds.get(FEEDID)

def delay(ms):
    time.sleep(1.0*ms/1000)

def setup():
    print "read channel ADC0 value ,the V-REF = 3.3V"
    delay(3000)

def loop():
    while True:
        value = analog_read(5)
        temp = value*(3.3/4096*100)
print ("value = %4d"%value)

print ("temperature = %4.3f V" %temp)

now=datetime.datetime.utcnow()

feed.datastreams=[ xively.Datastream(id='office_temp', current_value=temp, at=now)]

try:
    feed.update()
    print "Value pushed to Xively: " + str(temp)

except HTTPError as e:
    print "Error connecting to Xively: " + str (e)

time.sleep(20)

def main():
    setup()
    loop()

main()

To run the code:

$python sively-temp.py

We can see the data posted at xively.com webpage:
17. Control GPIO of pcDuino3 using web browser with Python (Python)

In this project, we will control the states of LEDs through a web browser. This project acts as the template of home automation.

**Install Software Tools:**

1. Update the apt-get.

   $ sudo apt-get update

2. Install Requests. Requests is the HTTP client library of the Python.

   $ sudo apt-get install python-requests

3. Install the python-pip, pip could replace the easy_install and manage the python software tool.
$ sudo apt-get install python-imaging python-imaging-tk python-pip python-dev

4. Install Flask. Flask is a low level Web application framework, use the Python to code.

$ sudo pip install flask

5. Install pcDuino-python sample library

$git clone https://github.com/pcduino/python-pcduino

Script:

1. Change the blink_led.py file under blink_led sample to web-led.py, and input the following code:

```python
import gpio

from flask import Flask, render_template, request

app = Flask(__name__)

pins = {
    'gpio0': {'name': 'Red_LED', 'state': False},
    'gpio1': {'name': 'Blue_LED', 'state': True}
}

for pin in pins:
    gpio.pinMode(pin, gpio.OUTPUT)
    gpio.digitalWrite(pin, gpio.LOW)

@app.route("/")

def main():
    for pin in pins:
        # code here
```
pins[pin]['state'] = gpio.digitalRead(pin)

templateData = {
    'pins':pins
}

return render_template('pin.html', **templateData)

@app.route("/<changepin>/<value>")
def action(changepin, value):
    setpin = changepin
    message = ""
    deviceName = pins[setpin]['name']
    if value == "on":
        gpio.digitalWrite(setpin, gpio.HIGH)
        message = "turned " + deviceName + " on."
    if value == "off":
        gpio.digitalWrite(setpin, gpio.LOW)
        message = "turned " + deviceName + " off."
    if value == "toggle":
        gpio.digitalWrite(setpin, not gpio.digitalRead(setpin))
        message = "toggled " + deviceName + "."

    for pin in pins:
        pins[pin]['state'] = gpio.digitalRead(pin)

    templateData = {
        'message': message,
        'pins': pins
    }

    return render_template('pin.html', **templateData)
if __name__ == "__main__":
    app.run (host='0.0.0.0',port=80,debug=True)

2. Build the templates file at another directory, and use the test editor to build the pin.html in the file, the code is as follows:

```html
<!DOCTYPE html>
<head>
    <title>Current Status </title>
</head>
<body >
    <center>
        <h1>Device Listing and Status</h1>
        <font size="5" color="red">
        <p>The {{ pins['gpio0'].name }}
        {% if pins['gpio0'].state == true %}
            is currently on !  (<a href="/{{ 'gpio0' }}/off">turn off</a>)
        {% else %}
            is currently off !  (<a href="/{{ 'gpio0' }}/on">turn on</a>)
        {% endif %}
        </font>
        </p>
        </font>
        <font size="5" color="blue">
        <p>The {{ pins['gpio1'].name }}
        {% if pins['gpio1'].state == true %}
          is currently on !  (<a href="/{{ 'gpio1' }}/off">turn off</a>)
        {% else %}
          is currently off !  (<a href="/{{ 'gpio1' }}/on">turn on</a>)
        {% endif %}
        </p>
    </center>
</body>
```

98 / 108
is currently off! (<a href="/{{ 'gpio1' }}/on">turn on</a>)

{% endif %}

</p>
</font>

{% if message %}
<h2>{{ message }}</h2>

{% endif %}

<hr>

<font size="6">

<a href="http://www.pcduino.org">pcDuino.org</a>

</font>

</center>
</body>
</html>

**Result:**

$ sudo python ./web-led.py

Open the browser on the another PC which share the same network of the pcDuino3, and input the IP address of the current pcDuino3, you will see the information as follows:
Open the browser on the another PC which share the same network of the pcDuino, and input
The current pcDuino IP address, you will see the information as follows:

Device Listing and Status
The Red_LED is currently on! (turn off)
The Blue_LED is currently off! (turn on)

turned Red_LED on.

pcDuino.org
18. Working with camera (SimpleCV)

18.1 Install SimpleCV

1. The steps to install SimpleCV:

   $ sudo apt-get install ipython python-opencv python-scipy
   $ sudo python-numpy python-setuptools python-pip
   $ sudo pip install https://github.com/ingenuitas/SimpleCV/zipball/master
   $ sudo apt-get install python-pygame
   $ sudo apt-get install python-imaging

2. After installed, input simplecv into shell to programing.

   ubuntu@ubuntu:~$ simplecv

   +-----------------------------------------------------------+
   + SimpleCV 1.3.0 [interactive shell] - http://simplecv.org +
   +-----------------------------------------------------------+

   Commands:

   "exit()" or press "Ctrl+ D" to exit the shell

   "clear" to clear the shell screen

   "tutorial" to begin the SimpleCV interactive tutorial

   "example" gives a list of examples you can run

   "forums" will launch a web browser for the help forums
"walkthrough" will launch a web browser with a walkthrough

Usage:

dot complete works to show library

for example: Image().save("/tmp/test.jpg") will dot complete

just by touching TAB after typing Image().

Documentation:

help(Image), ?Image, Image?, or Image()? all do the same

"docs" will launch webbrowser showing documentation

SimpleCV:1> cam=Camera()

VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument
VIDIOC_QUERYMENU: Invalid argument

SimpleCV:2> img=cam.getImage()

SimpleCV:3> img.show()

SimpleCV:5:
A picture captured will be displayed on the screen:

18.2 Basic Camera function with SimpleCV on pcDuino3

After we install SimpleCV, we can copy and paste the following python code:

```python
from SimpleCV import Camera, Display
from time import sleep
myCamera = Camera(prop_set={'width': 640, 'height': 480})
myDisplay = Display(resolution=(640, 480))
while not myDisplay.isDone():
    myCamera.getImage().save(myDisplay)
sleep(.1)
```
Save it to a file named 'basic-cam.py', run it using '$python basic-cam.py':

![Image](image_url)

### 18.3 Detect face with SimpleCV on pcDuino3

In this project, we will try to detect face. The python is as below:

```python
from SimpleCV import Camera, Display
from time import sleep

myCamera = Camera(prop_set={'width':640, 'height': 480})

myDisplay = Display(resolution=(640, 480))

while not myDisplay.isDone():
    frame = myCamera.getImage()

    faces = frame.findHaarFeatures('face')

    if faces:
        for face in faces:
```

```
print "Face at: " + str(face(coordinates))

else:

    print "No faces detected."

frame.save(myDisplay)
sleep(.1)

We copy and paste the above code and save into a script. Run the python script, we will be able to detect the face that is present in the camera:
19. Android Smart TV Box

In this project, we will reflash pcDuino3 with an Android operation system. We use Win32diskimagewrite to create a production SD card, insert the SD card into pcDuino3’s SD slot, and reboot pcDuino3 with the SD in. There will be no LED activities to indicate the progress. Please wait for 10 minutes. Remove the SD card, and power cycle pcDuino3 again. We will see Android booting.

The following picture shows a pcDuino3 running Android 4.2 hooked to a HDMI TV. With a wireless mouse/keyboard or an air mouse. We can install APPs from google play store, and start to watch movies using APPS like youtube and Netflix.
An app called PPTV playing Chinese TVs:

With the installation of Airplay app, it can become an apple TV or google chrome cast.
我是个新丁，师兄说什么我便做什么