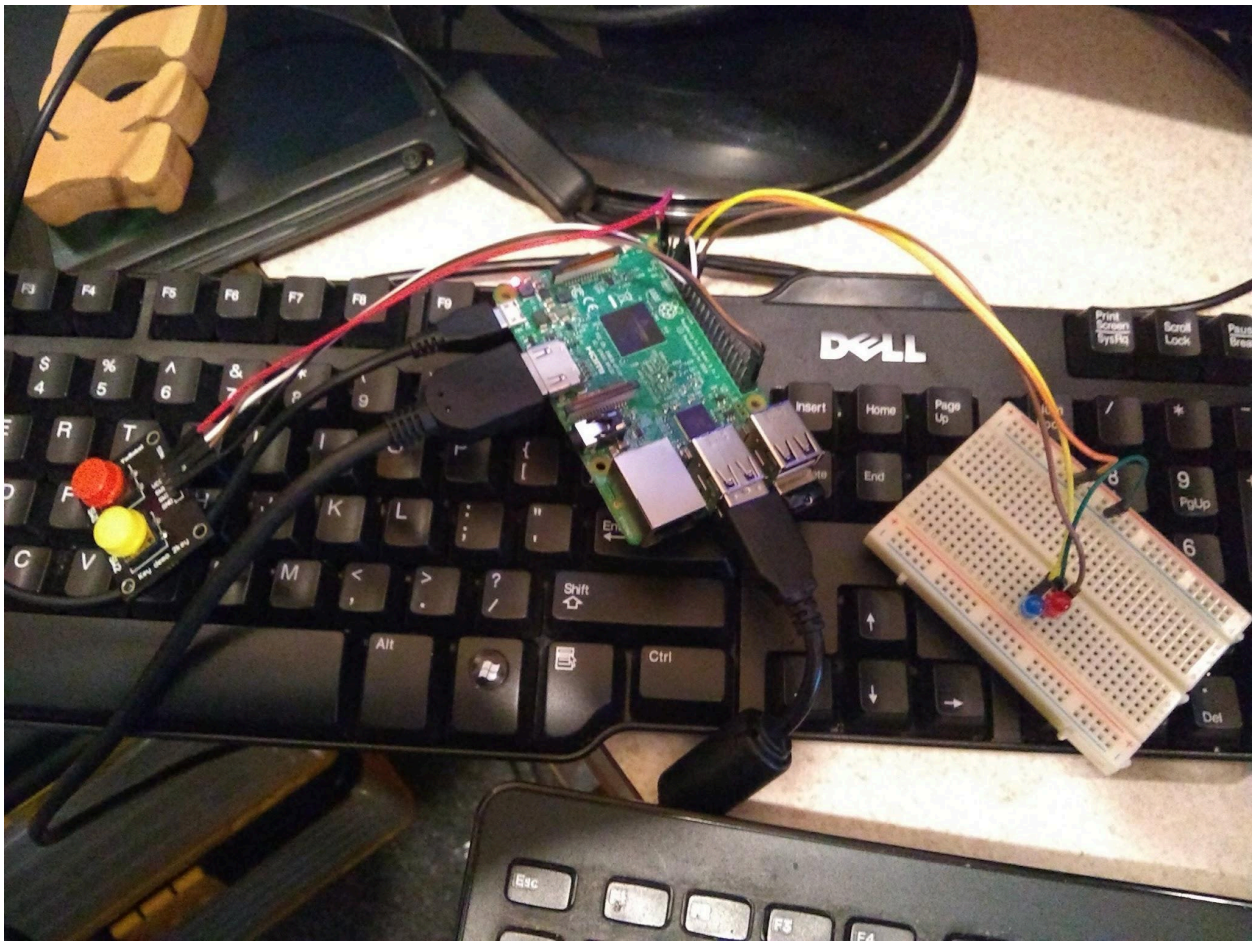
Programming Raspberry PI with C

Using the Arduino IDE, Particle Cloud, and Pure C

Tested with a Raspberry PI 3, using Raspberrian Jessie (and Raspberrian Jessie Lite).
I've been able to flash LEDs and use push buttons to turn LEDs on and off.



Test Sketches can be found:
https://github.com/automation-technology-club/RaspberryPI-C-Programming

Arduino IDE:

To program the Raspberry PI with the Arduino IDE takes some setup, each step is important - missing a step may or may not cause issues. I'll go over the steps that I did.
You can find a lot more information and support on the project Github page.
https://github.com/me-no-dev/RasPiArduino

Features

- The familiar Arduino API
- pinMode/digitalRead/digitalWrite/analogWrite
- Full SPI, Wire and Serial compatibility
- Access to STDIN/STDOUT through the Console class
- Access to system tty through the TTY library
- Process, FileIO, Client, Server and UDP implementations through the Bridge library

We need to setup the IDE on a desktop computer - the instructions on the github site are pretty straight forward for this setup. It boils down to cloning the github repository into your Arduino IDE hardware directory - and installing the correct tool chain for your OS.
There ended up being a couple of things to note about this:

1) Your Raspberry PI will not show up in the “Ports” list until it is setup.
2) And The board listed is the Raspberry PI 2+

The 2nd might throw you off if you are using a different model - it seems to work with all the current boards (Pi Zero, Pi Zero W, Pi 2 and Pi 3)

Next we need to setup the Raspberry PI - The recommended OS is Jessie.
There are a number of steps here. Some of these can be done using the raspi-config, and some are already turned off by default.
It is important to follow the steps closely. What is going on is we are enabling the root user, and setting it up to be used as the user to run the Arduino Sketches.
(There are a few optional steps as well - like setting a host name for the PI, etc)
It should be noted that WIFI is listed as optional (I think this is because you can use the ethernet port on the PI) - you probably want this wireless thou.

NOTES: I had an issue with disable Serial Console step - this is because my install has 3 boot partitions and three different OSes running on it. (I used the raspi-config utility to turn off the Serial console)

And the Avahi service is not OPTIONAL and needs to be setup.

NOTE: My new router gives a IPv6 address, and this seems to cause problems with getting the time synch with the PI.
To force a IPv4 do this:
Raspbian Jessie no longer has a file called `/etc/sysctl.d` -- it's now a ***folder*** of the same name.

According to the read-me file in the same folder, `README.sysctl`, it basically says that any `.conf`file will be read in at boot time and processed. Any legal file name will work, but they suggested `local.conf`, so that's what I used.

I created `/etc/sysctl.d/local.conf` and added the single line from the other answers:

```
net.ipv6.conf.all.disable_ipv6=1
```

This appears to work just fine.

https://raspberrypi.stackexchange.com/questions/23884/force-ipv4-address

Once you get everything setup, reboot the PI. And if all went well you should see the IP address of the PI in the PORT of the Arduino IDE.
NOTES: I had to refresh the network on my desktop machine to see the Raspberry PI - I think this has to do with my router, and not the setup of this.

I have a few example sketches that I have working in our github repository. (https://github.com/automation-technology-club/RaspberryPI-C-Programming)

More NOTES: This appears to use the BCM_GPIO numbers for the GPIO pins on the Raspberry PI.

Even More NOTES: The sketch that is uploaded to the Raspberry PI is compiled, it can be copied and executed as needed. (CTRL-C exits most of the time, and SUDO is needed to execute the sketch)
The application is in the */usr/local/bin* directory. It is the “arduino-sketch” file, all you need to do is copy this file to your home directory. Like: *cp arduino-sketch ~/anewnameforsketch*
Then you just need to issue a *sudo ./anewnameforsketch* command in your home directory and your sketch should run.

Want Even More Notes: I have mine set to autoboot and load the sketch - I noticed that if I run a app that I saved (a compiled app), that the autoboot sketch is also running. This is apparent when using the PWM on pin 18. I think the only way to stop the sketch is to look at TOP and kill the PID.

Particle Cloud Setup:

***** The Raspberry PI with Particle Cloud is currently in beta *****

A question that comes up at the meetings is why would you use Particle Cloud - I think I have a answer - actually I think I have two answers.

1) For an Individual using a service like Particle let's you update the firmware of remote devices. They don't need to be connected to the same network you are.
2) For Companies (or products) - it's a good way to update firmware across many devices at once, (They have corrupted accounts who make microcontrollers for the Particle Cloud).

Why use Particle - For the most part the IDE is very much like the Arduino IDE - so it's something with a small learning curve if you do program with Arduino IDE.
(NOTE: There are some differences, but for the most part they are more similar than they are different)
Particle also has an API for interactive via a web-app, or a REST client to control your micro-controller.

***** The Raspberry PI with Particle Cloud is currently in beta *****

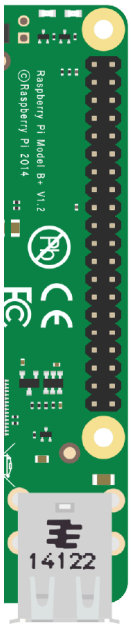
https://docs.particle.io/guide/getting-started/start/raspberry-pi/
Setup for this is much easier, you do need a Particle Cloud (https://particle.io) account.
And setup is only done on the Raspberry PI, it does take a little while, but the process is automated. And when it's done you'll have a Raspberry PI show up in your Particle devices
You will want at least Raspberry PI 2, and running Jessie. (I did try this with a PI Zero W, it worked, so I think this will work on all flavors of PI)

Run this line from the CLI of the Raspberry PI.
`bash <( curl -sL https://particle.io/install-pi )`

That's it.

Particle is also very good about documentation:
https://docs.particle.io/datasheets/kits-and-accessories/raspberrypi-datasheet/

Something to note: They do have there own PIN number skeem



Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
3.3V			1	X	X	2	5V		
I2C	GPIO2	SDA	3	X	X	4	5V		
	GPIO3	SCL	5	X	X	6	GND		
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
GND			9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14	GND		
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
3.3V			17	X	X	18	D11/A2	GPIO24	Digital I/O
SPI	GPIO10	MOSI	19	X	X	20	GND		
	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
GND			25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30	GND		
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34	GND		
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
GND			39	X	X	40	D16/A7	GPIO21	Digital I/O

I found that they also support the GPIO number skeem.
(Another Note: Raspberry PI has no Analog input, even thou Particle has some PINs listed with a Analog reference number)

Particle Starts the Sketch each time the PI is rebooted, I haven't been able to find where it saves the compiled executable.

"Pure C"

If you want something a lot closer to "Pure C", there is a wonder library call "wiringPi".
https://projects.drogon.net/raspberry-pi/wiringpi/pins/
http://wiringpi.com/

Any text editor can be used to write your programs. A "C" compiler should already be installed by default in Jessie.
To Install wiringPi - follow these instructions:
https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/

A simple "Blink LED" C program:

```
#include <wiringPi.h>
#include <stdio.h>

#define LedPin 0

int main(void) {
        if(wiringPiSetup() == -1) { //when initialize wiringPi failed, print
message to screen
                printf("setup wiringPi failed !\n");
                return -1;
        }

        pinMode(LedPin, OUTPUT);
        while(1) {
                digitalWrite(LedPin, LOW);   //led on
                printf("led on\n");
                delay(1000);                  // wait 1 sec
                digitalWrite(LedPin, HIGH);  //led off
                printf("led off\n");
                delay(1000);                  // wait 1 sec
        }
        return 0;
}
```

A few things to note here - 1st you'll notice there is no void loop() { } This is because this is C not Arduino. Next thing you might notice is the PIN number is 0, This is because WiringPi uses its own PIN number skeem, that can be found here: http://wiringpi.com/pins/ So wiringPi PIN 0 is GPIO Pin 17 - this can be confusing, so refer to the wiringPi pins table when using "C".

To compile this program to run on the command line (CLI) of the raspberry pi type:
gcc -o blink blink.c -lwiringPi
The flag -o is for the output name of the compiled file, blink.c will be the source and the -lwiringPi is to include the wiringPi library (This is a lower case L, not an I)

To run the new blink we type:
sudo ./blink
And if you have a LED hooked up to the correct PIN you see it blinking.

*** A Really special NOTE: PWM can be used, however I found that if you make some mistakes in your program real bad things can happen. I had my SD unmount, and I had some programs get destroyed by making a small mistake while I was compiling a C program that used PWM ***

Another note about PWM - The Raspberry PI only has 4 PWM pins - There is a software PWM included with the wiringPi library (I haven't tried it). To use PWM you need to do this:
pinMode(1, PWM_OUTPUT); //GPIO 18

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Board Revisions: Please note the differences between board revisions 1 and 2 (Rv1 and Rv2 above) The Revision 2 is readily identifiable by the presence of the 2 mounting holes.

P5: Secondary GPIO connector (Rev. 2 Pi only)							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		5v	1	2	3.3v		
17	28	GPIO8	3	4	GPIO9	29	18
19	30	GPIO10	5	6	GPIO11	31	20
		0v	7	8	0v		
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Update: Dec 20, 2017

1) Demo now has all three “demos” on one Raspberry PI install/partition (no need to reboot the PI).
    a) PiDuino can be killed using “TOP” and “sudo kill PID#”
    b) Particle can not be killed using “sudo kill PID#” it spawns again
        i) To stop a sketch upload a “empty.ino” file this seems to work.
2) Uploading an EMPTY.ino appears to reset the PINs and I/O for something else to work
3) Could not get DHT11 to work with Particle, didn’t try hard however.
4) PiDuino sketches can be found at /usr/local/bin - the compiled sketch is “arduino-sketch”
5) PiDuino sketches need to be run with SUDO.
6) Some of wiringPi sketches have to be used with SUDO - others don’t (?)
7) To run a sketch in the background add a & to the command line IE: “sudo ./blink &” will run the blink sketch in the background
8) To run a sketch in the background without Console output add >/dev/null & IE: “sudo ./blink >/dev/null &” This will pipe the output of the sketch to the null device and run it in the background.
9) To kill a background task use “TOP” and “sudo kill PID#”

Added a DHT11 to pin 23 (wiringPi Pin 4) - the C and Arduino have been added to the github repository. More Information can be found here:
http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/
But I only used part of this.

The Arduino sketch is an old sketch/library that can be found on the Arduino.cc website - it was only modified for the pin number of the PI and to change “Serial.print” to “Console.print”

- Special note: Piduino has replaced Serial with Console it’s almost a direct drop in replacement, just use “find and replace”
- Particle uses “Particle.publish” to see the “serial” output on the website - It is NOT a direct drop in replacement and takes some extra steps to get it working. Particle also supports “Serial” - but it looks like you need to have the microcontroller hooked up to the computer to see the output.

My Arduino DHT11 sketch compiles once - uploads and if changes are made they need to be saved and the sketch needs to be reopened - for it to compile again (Yes that is strange) - I googled the error, and it seems to be somewhat common for boards that aren’t 100% supported by the IDE. (The work around is save, reload sketch, upload again - no one seems to be working on the problem to fix it however).

Update: May 9, 2018 Yet another way to program the PI in C.

https://github.com/NVSL/PiDuino_Library
PiDuino Library - the repo says it's be depreciated, but it currently still works. This is close to "Arduino C", works a bit like WiringPI but a little easier to use, since it's a bit closer to Arduino.

Update: May 15, 2018 Arduino Create Online

So, it looks like you can use the online Arduino IDE (Create)
https://create.arduino.cc/
https://www.techrepublic.com/article/raspberry-pi-powered-by-arduino-now-you-can-program-the-pi-using-arduino-sketches/
I found this to be a bit clumsy to use, and didn't find much information about using it.
Use the GPIO numbers.