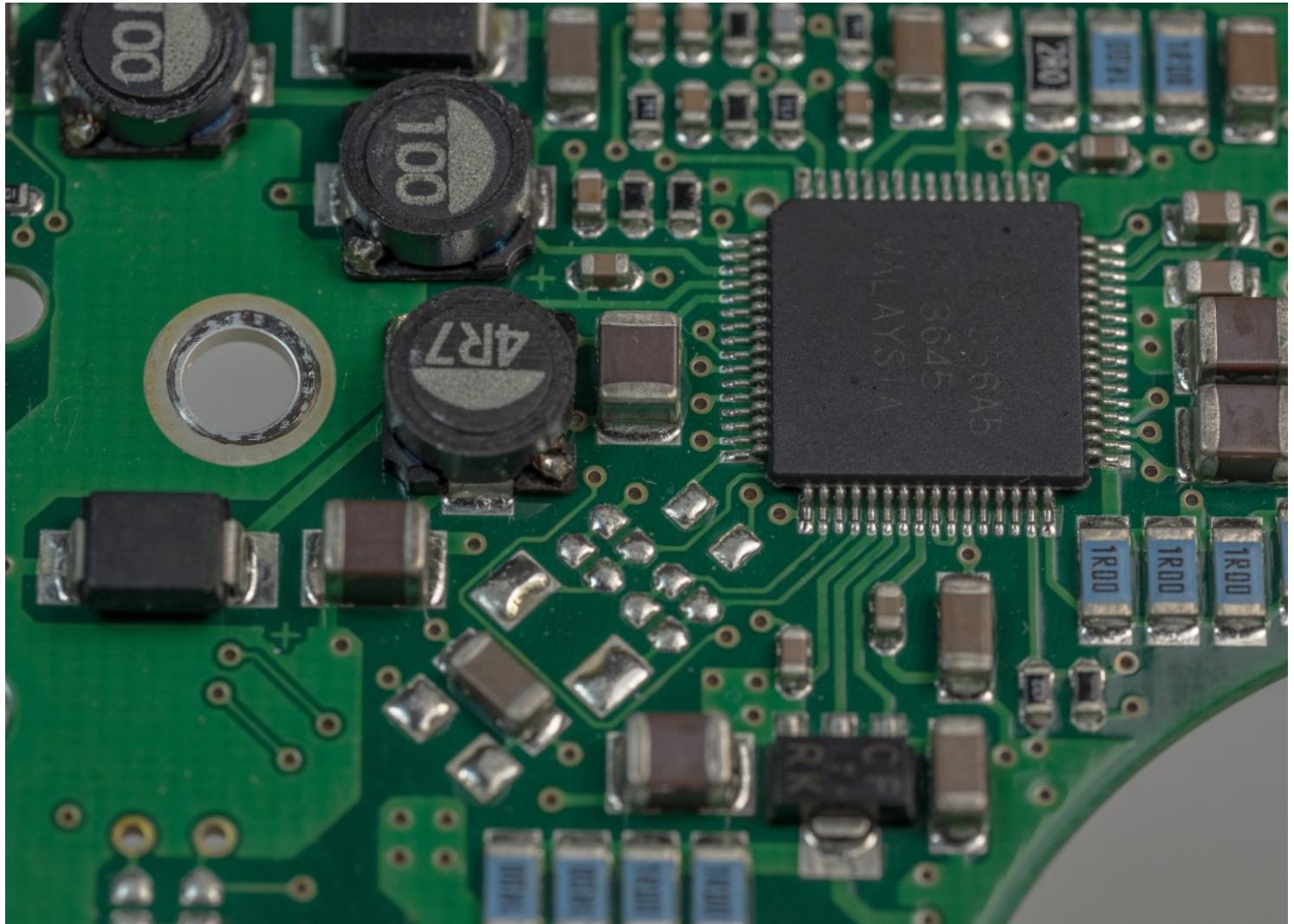


[HOME](#) [ABOUT](#) [BLOG](#)

Tutorial: How to Design Your Own Microcontroller Board – I

45 Comments



Article Technical Rating: 7 out of 10

This is the first in a series of tutorials where you'll learn how to design your own custom microcontroller board. To get the most out of this tutorial it's critical that you read the article and watch the included video.

Initially, we're going to focus on just the microcontroller itself so you can more easily understand the design process without getting overwhelmed with circuit complexity.

I'll break down the design process into three fundamental steps:

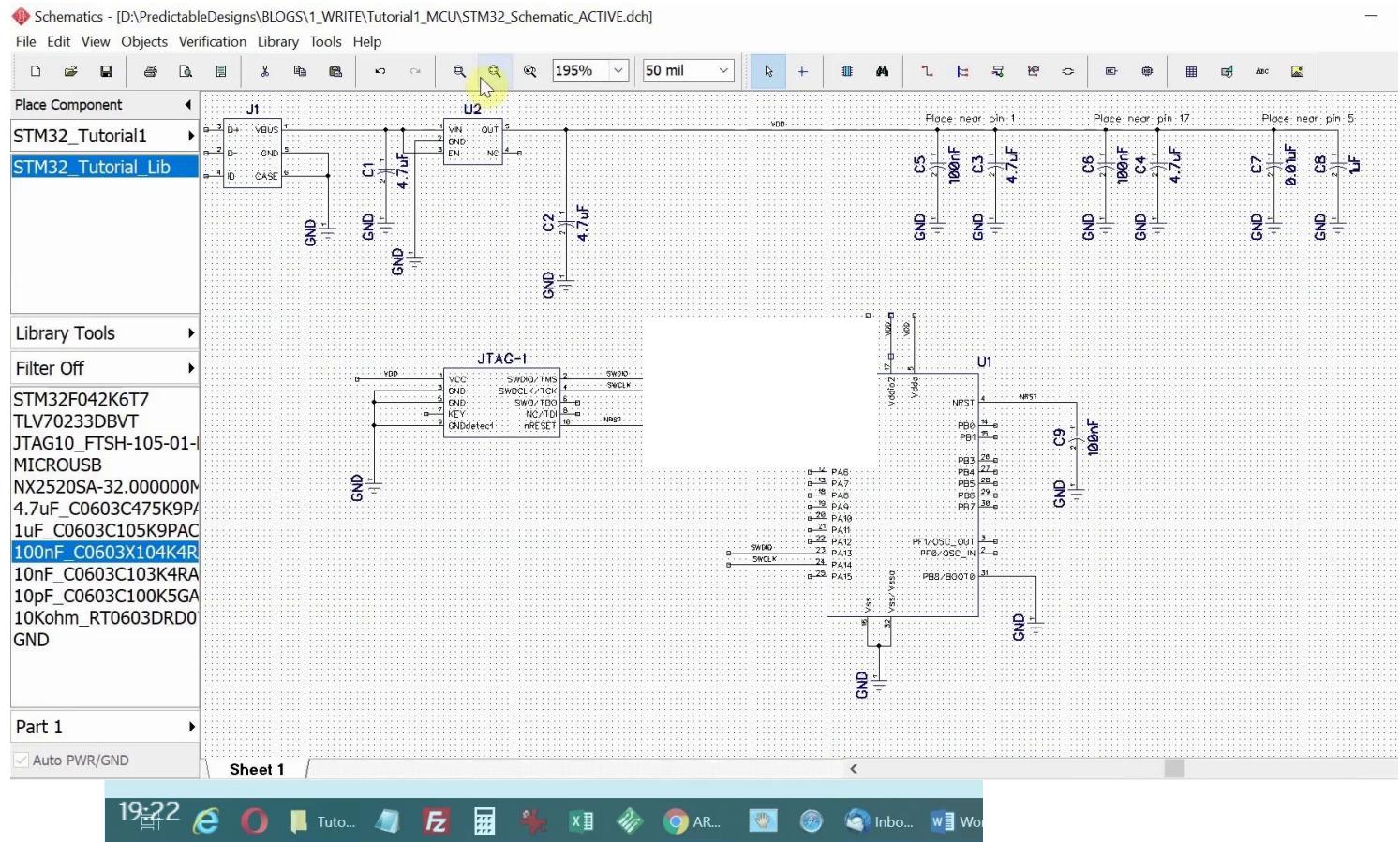
STEP 1 – System Design

STEP 2 – Schematic Circuit Design

STEP 3 – PCB Layout Design

You're going to learn how to design the system and the schematic circuit in this first tutorial. Then, in part two you'll learn how to lay out the Printed Circuit Board (PCB) and order prototypes.

NOTE: This is a long, very detailed tutorial so here's a [free PDF version](#) of part 1.



If you like this tutorial then I suggest you join my upcoming **Hardware Academy** program. This program will provide you with regular in-depth training sessions (on both technical and entrepreneurial topics) with me and other experts. Plus you get direct access to me for support.

questions and advice. To get early access and discounted pricing [join the waiting list](#) to

System / Preliminary Design

When developing a new circuit design the first step is the high-level system design (which we also call a [preliminary design](#)). Before getting into the details of the full schematic circuit design it's always best to first [focus on the big picture](#) of the full system.

Designing the system consists mainly of two steps: creating a block diagram and selecting the critical components (microchips, sensors, displays, etc.). A system design treats each function as a black box

In engineering, a black box is an object which can be viewed in terms of its inputs and outputs but without any knowledge of its internal workings. With a system-level design focus is on the higher level interconnectivity and functionality.

Block Diagram

Below is the block diagram that we'll be working from in this tutorial series. As I mentioned for this first tutorial we'll focus just on the microcontroller itself. In future tutorials we'll expand the design to include all of the functionality shown in this block diagram.

A block diagram should include a block for each core function, the interconnections between the various blocks, specified communication protocols, and any known voltage levels (supply voltage, battery voltage, etc.).

Later, once all of the components have been selected and the required supply voltages are known I like to add the supply voltages to the block diagram. Including the supply voltages for each functional block it allows you to easily identify all of the supply voltages you'll need as well as any level shifters.

In most cases when two electronic components communicate they need to use the same supply voltage. If they are supplied from different voltages then you'll usually need to add a level shifter.

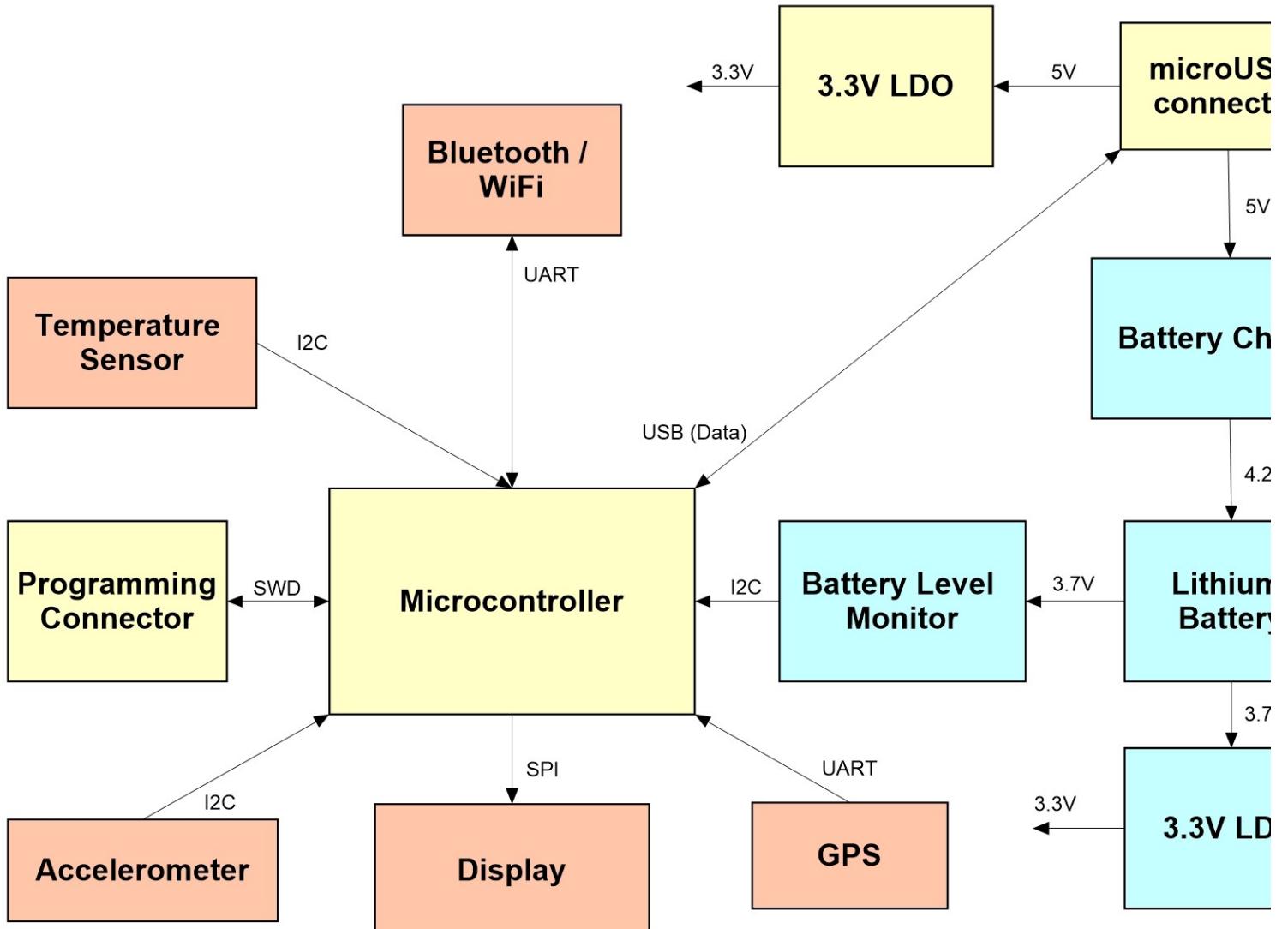


Figure 1 – System-level block diagram. Blocks in yellow are included in this initial tutorial. Other blocks/functions will be added in future tutorials.

Now that we have a block diagram we can better understand the necessary requirements for the [microcontroller](#). Until you've mapped out everything that will connect to the microcontroller it's impossible to select the appropriate microcontroller.

Select Microcontroller

When selecting a microcontroller (or just about any electronic component) I like to use an electronics distributor's website like [Newark.com](#). Doing so allows you to easily compare various options based on a variety of specifications, pricing, and availability. It's also a great way to quickly access the component's datasheet.

If you regularly read this blog you'll know that I'm a big fan of ARM Cortex-M microcontrollers. Arm Cortex-M microcontrollers are easily the most popular line of microcontrollers used in commercial electronic products. They have been used in tens of billions of devices.

Microcontrollers from Microchip (including Atmel) may dominate the maker market but Arm dominates the commercial product market.

Arm doesn't actually manufacture the chips directly themselves. They instead design processor architectures that are then licensed and manufactured by other chip makers including ST, NXP, Microchip, Texas Instruments, Silicon Labs, Cypress, and Nordic.

The ARM Cortex-M is a 32-bit architecture that is a fantastic choice for more computationally intensive tasks compared to what is available from older 8 bit microcontrollers such as the 8051, PIC, and AVR cores.

Arm microcontrollers come in various performance levels including the Cortex-M0, M1, M3, M4, and M7. Some versions are available with a Floating Point Unit (FPU) and are designated with an F in the model number such as the Cortex-M4F.

One of the biggest advantages of Arm Cortex-M processors is their low price for the level of performance you get. In fact, even if an 8-bit microcontroller is sufficient for your application you should still consider a 32-bit Cortex-M microcontroller.

There are Cortex-M microcontrollers available with very comparable pricing to some older 8-bit chips. Basing your design on a 32-bit microcontroller gives you more room to grow should you want to add additional features in the future.

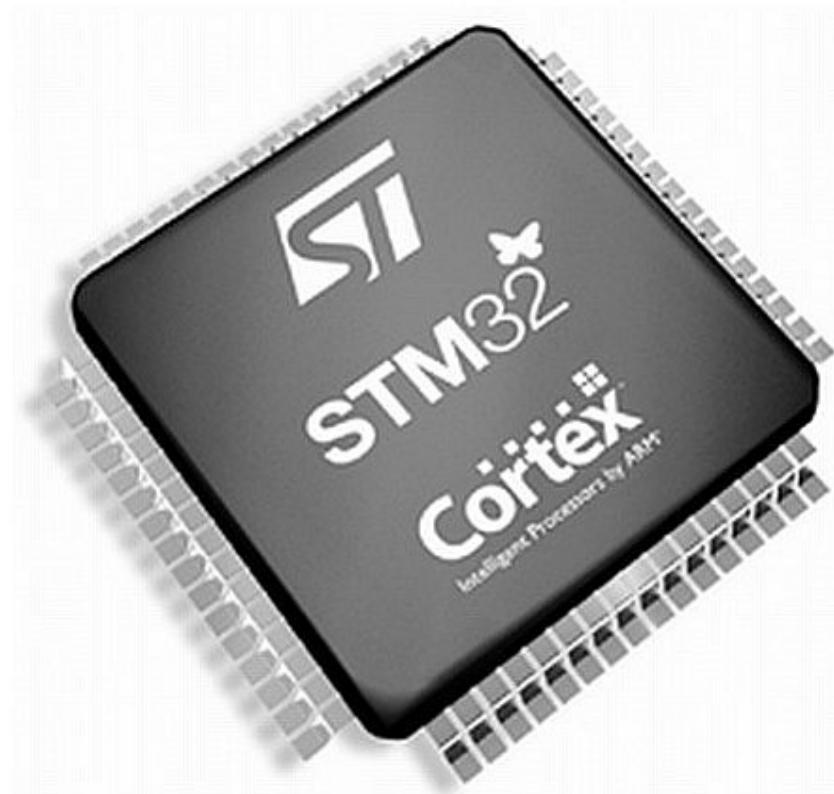


Figure 2: The STM32 from ST Microelectronics is my favorite line of ARM Cortex-microcontrollers.

Although numerous chip makers offer Cortex-M microcontrollers, my favorite by far is the STM32 series from ST Microelectronics. The [STM32](#) line of microcontrollers is quite expansive with just about any feature and level of performance you would ever need. The STM32 line can be broken down into several subseries as shown in Table 1 below.

STM32 Series	Cortex-Mx	Max clock (MHz)	Performance (DMIPS)
F0	M0	48	38
F1	M3	72	61
F3	M4	72	90
F2	M3	120	150
F4	M4	180	225
F7	M7	216	462
H7	M7	400	856
L0	M0	32	26
L1	M3	32	33
L4	M4	80	100
L4+	M4	120	150

Table 1: Comparison of various STM32 microcontroller variants

The STM32F subseries is their standard line of microcontrollers (versus the STM32L subseries which is specifically focused on lower power consumption). The STM32F0 has the lowest price but also the lowest performance. One step up in performance is the F1 subseries, followed by the F3, F2, F4, F7, and finally the H7.

For this tutorial I have selected the STM32F042K6T7 which comes in a 32-pin LQFP leaded package. I selected a leaded package primarily because it simplifies the debugging process because you have easy access to the microcontroller pins. Whereas with a leadless package like a QFN, the pins are hidden away underneath the package making access impossible without test points.

A leaded package also allows you to more easily swap out the microcontroller if it were to become damaged. Finally, leadless packages cost more to solder on to the PCB so they increase both the prototyping and [manufacturing costs](#).

I selected the STM32F042 because it offers moderate performance, a good number of pins, and various serial protocols including UART, I2C, SPI and USB. This is a fairly entry level STM32 microcontroller with only 32 pins, but with a wide variety of features. More advanced versions come with as many as 216 pins which would be quite overwhelming in an introductory tutorial.

In this first video we won't be using most of these features, but we will take advantage of them in future videos in this series.

Schematic Circuit Design

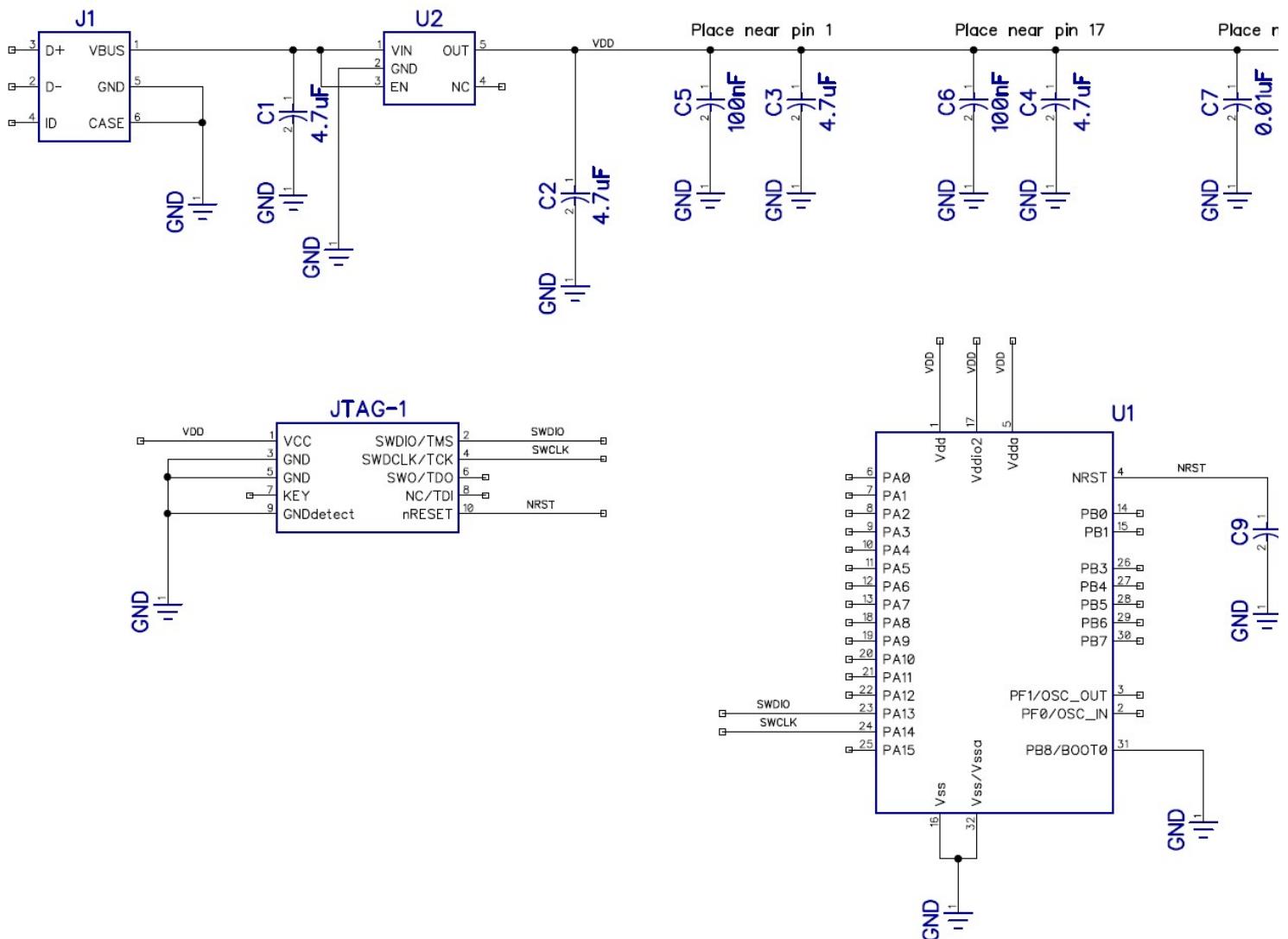


Figure 3: Schematic circuit diagram for this first tutorial showing the STM32 microcontroller, linear regulator, USB connector, and programming connector.

Now that we have selected the microcontroller it's time to design the schematic circuit diagram. For these tutorials I'll be using a PCB design tool called [DipTrace](#).

There are dozens of PCB tools available but when it comes to ease of use, price, and performance I find that DipTrace is hard to beat, especially for beginners.

If you don't have a PCB design package then you may want to consider downloading the free version of DipTrace so you can follow along closely with this tutorial. The best way to learn something is always to actually do it.

For this initial tutorial the free version of DipTrace is sufficient, but as we expand the functionality of this circuit in future tutorials you'll need to upgrade to a paid version of DipTrace or their 30-day free trial.

DISCLOSURE: I'm an affiliate for DipTrace, but only after I used it to successfully design dozens of products. I've been using it now for almost 4 years, and I still love it. If you do decide to purchase DipTrace, I would really appreciate it if you would [use this link](#) so I receive a commission at no extra cost to you.

Nonetheless, this tutorial will be focusing on the process for designing a custom microcontroller board, and not on how to use any specific PCB design tool. So regardless of which PCB software you end up using you'll still find these tutorials are just as useful.

The first step in designing a schematic is to place all of the key components. For this initial design this includes the microcontroller chip, a voltage regulator, a microUSB connector, and a programming connector.

For more complex designs it usually makes more sense to completely design each sub-circuit first, then merge them all together. Depending on the design complexity (and personal preference) you may also want to place each sub-circuit on its own separate sheet. This keeps the schematic from becoming a huge, overwhelming monster on a single sheet.

Capacitors

Next, we'll place all of the various capacitors. For the most part you can think of capacitors as tiny little rechargeable batteries that hold electrical charge and help to stabilize the voltage on a supply line.

We'll start by placing a 4.7uF capacitor on the input pin of the linear regulator. This is the 5VDC input voltage supplied by an external USB charger. This voltage is fed into a TLV70233 linear regulator which steps the voltage down to 3.3V since the microcontroller can only be supplied by a maximum of 3.6V.

Another 4.7uF capacitor is placed on the output of the regulator as close to the pin as possible. This capacitor serves to store charge to supply transient loads and it acts to stabilize the internal feedback loop of the regulator. Without an output capacitor most regulators will begin to oscillate.

Decoupling capacitors must be placed as close as possible to the microcontroller supply (VDD). It's always best to refer to the microcontroller datasheet in regards to their recommendations for decoupling capacitors.

The datasheet for the STM32F042 recommends a 4.7uF and a 100nF capacitor be placed next to each of the two VDD pins (input supply pins). It also recommends 1uF and 10nF decoupling capacitors be placed near the VDDA pin.

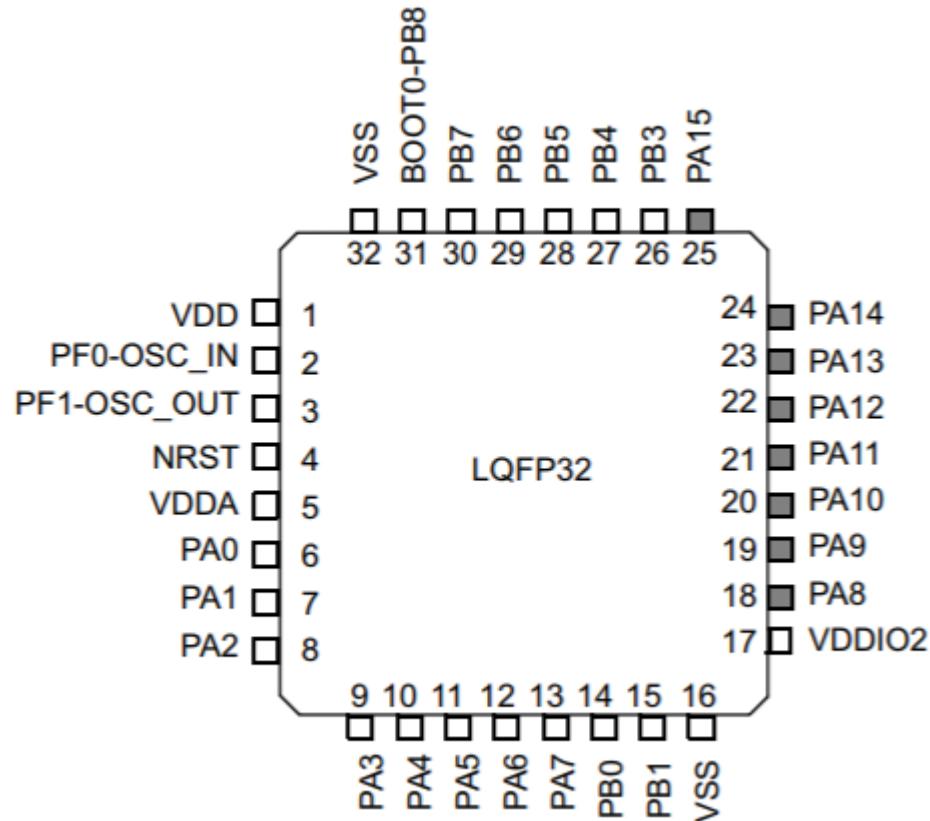
The VDDA pin is the supply for the internal analog-to-digital (ADC) converter and must be especially clean and stable. We're not using the ADC in this first tutorial but we will in a future one.

Note that you'll commonly see two capacitor sizes specified together for decoupling purposes. For example, 4.7uF and 100nF capacitors.

The larger 4.7uF can store more charge which helps stabilize the voltage when large spikes in load current are required. The smaller capacitor serves mainly to filter out any high-frequency noise.

Microcontroller pinout

Although the STM32F042 offers a wide variety of functions such as UART, I2C, SPI, and communication interfaces, you won't find any of these functions labeled on the microcontroller pinout. This is because most microcontrollers assign a variety of functions to each pin so as to reduce the number of pins required.



Refer to the pin definition table in the microcontroller datasheet (page 33 for the STM32F042) which shows all of the various functions available for each pin. Always be sure to confirm that two functions required for your product don't overlap on the same pins.

Clock

All microcontrollers require a clock for timing purposes. This clock is just an accurate oscillator. Microcontrollers execute programmed commands sequentially with each tick of the clock.

The simplest option, if available on the selected microcontroller, is to use the internal clock. This internal clock is known as an RC oscillator clock because it uses the timing characteristics of a resistor and capacitor.

The major downside to an RC oscillator is accuracy. Resistors and capacitors (especially those embedded inside a microchip) vary significantly from unit to unit causing the oscillator frequency to vary. Temperature also significantly impacts the accuracy.

An RC oscillator is fine for simple applications, but if your application requires accurate timing then it won't be sufficient. For this initial tutorial we're going to use the internal clock to keep things simple. In future tutorials we'll improve the design by adding a much more precise, external crystal-based oscillator.

Programming Connector

Programming an STM32 is done via one of two protocols: JTAG or Serial Wire Debug (SWD). More advanced versions of the STM32 (STM32F1 and higher) offer both JTAG and SWD programming interfaces. The STM32F0 subseries offers only the simpler SWD programming interface so that is what we will focus on for this tutorial.

The SWD interface requires only 5 pins. They are SWDIO (data input/output), SWCLK (clock signal), NRST (reset signal), VDD (supply voltage) and ground.

Unfortunately the ST-LINK programmer device that you'll use to program the STM32 has a 20-pin JTAG connector (with SWD functionality). This connector is quite large and is not practical for smaller board designs.

Instead, you can use a 20-pin to 10-pin adapter board such as [this one from Adafruit](#) so you can use a smaller 10-pin connector on your board.

For this tutorial we will use the 10-pin connector. If that is still too large for your project, then you can always use a 5-pin header and jumper wires from the 20-pin programmer output to connect only the 5 lines required for SWD programming.

Power

The last part of the schematic we'll cover is the power section. The STM32 microcontroller can be powered with a supply voltage from 2.0 to 3.6V. Unless you have a variable power supply, you'll need an on-board regulator to provide the appropriate supply voltage.

For this design we'll power the board using an external USB charger which outputs 5 V. This voltage will then feed into a linear voltage regulator (TLV70233 from Texas Instruments) which steps it down to a stable 3.3V.

The STM32 requires a maximum of only 24mA assuming none of the GPIO pins are sourcing any current (each GPIO pin can source up to 25mA). The absolute maximum current the STM32 will ever require is 120mA assuming various GPIO pins are sourcing current.

The TLV70233 is rated for up to 300mA which should be more than sufficient for this initial design. In future tutorials, as we add additional functions, we may need to revisit this to ensure the regulator can handle the required system current.

Electrical Rules Check

The final step of designing the schematic circuit diagram is to perform a verification step called an Electrical Rules Check (ERC). This verification step checks for errors such as shorts between nets, nets with only one pin, superimposed pins, and unconnected pins.

You can also setup various pin type errors. For example, if an output pin is connected to another output you will get an error. Or if an output pin is connected to a power supply you will get an error. DipTrace uses a colored grid matrix that allows you to define which type connections will give you errors or warnings.

Summary

In this first tutorial we've designed the system block diagram, selected all of the critical components, and designed the full schematic circuit diagram.

In [part 2 of this tutorial series](#) we'll focus on designing the actual printed circuit board layout and ordering board prototypes.

Do you need *affordable coaching, training, and support* to bring your new electronic hardware product to market? If so, join the [waiting list for the Hardware Academy](#) and early access with discounted pricing.

45

[Leave a Comment](#)

b i link b-quote u ul ol li code spoiler



Join the discussion...

[Subscribe ▾](#)[▲ newest](#) [▲ oldest](#) [▲ mc](#)

Elliott

Great presentation John. Thanks

Guest

+ 0 -

[Reply](#)

John Teel

Thank you Elliot!

Author

+ 0 -

[Reply](#)

Sumeet

You are awesome John !!!

Guest

+ 1 -

[Reply](#)

John Teel

Thanks Sumeet!

Author

+ 0 -

[Reply](#)

mahesh

hi John Teel ,superb..

i would like to design PCB which software is good.and online free software is it available?

Guest

+ 0 -

[Reply](#)

John Teel

DipTrace has a free version that is good up to 300 pins and 2 PCB layers.

Author

+ 0 -

[Reply](#)



Guest

ヴィトンバッグスピーディ

whoah this blog is fantastic i really like reading your posts.

Keep up the great work! You know, many persons are hunting around for this information, you can aid them greatly.

+ 0 -

[Reply](#)

John Teel

Author

Thank you!

+ 0 -

[Reply](#)

Guest

Kevin Vo

Thank you John for the great content,
When is part 2 and 3 come out?

+ 0 -

[Reply](#)

John Teel

Author

Thanks Kevin, I'm glad it was helpful. Part 2 is already published [here](#). I'm not sure yet when part 3 will come out, so please keep checking back in.

+ 0 -

[Reply](#)

Guest

paul

hello

Thank you for this article . I am new to micro controllers and found this article very well written and in a language I could understand. After looking at many websites and feeling like I was going in circles and swamped by technical specs and jargon . I found your article and learnt more in 15mins than I had in hours. I like the ground up approach of your tutorials.

+ 0 -

[Reply](#)

John Teel

Author

Hi Paul, that is so awesome to hear!! I do enjoy explaining complex topics in a way that people can act understand. So thank you for this comment.

John

+ 0 -

[Reply](#)

**Brendan**

Guest

Thank you so much for this tutorial! It's what my electrical engineering degree program was missing, hone Regarding the Power Scheme for "2 x VDD", I saw it said "2 x 100 nF + 1 4.7 uF". I think I understand it corr there are 2 VDD pins on opposite sides of the chip, they both require a 100 nF and 4.7 uF cap separately. However order of operations (2 x large cap+1 x small cap) and the drawing leave a lot up to interpretation. you please clarify how I should read that? Thank you.

+ 0 -

[Reply](#)

**John Teel**

Author

Thank you Brendan, I'm glad it was helpful. On the particular MCU version I've used there is one VDE one VDDIO pin (for I/O). Each of those pins should have a 100nF and a 4.7uF capacitor, although if ea tied to the same supply voltage you could also just do a 100nF on each pin and a single 4.7uF that is sl between them. I hope that clarifies your question, if not just let me know.

John

+ 0 -

[Reply](#)

**Vimal Kumar CR**

Guest

Hi John teel.

Thank you very much for helping all the electronics hungry people around the globe. Your data and article great frame for my startup.

Thank you once again. Gratitude !!

+ 0 -

[Reply](#)

**John Teel**

Author

You are most welcome.

Thanks!

+ 0 -

[Reply](#)



Taofeek

Your articles are always great and light up my Hardware career. Thanks for the great work. Also I saw a guy that program STM32 directly USB without St link. Also some people program STM32 series in Arduino IDE. Is there any best IDE to program STm. Because I know some people use open system workbench, True studio, Keil etc

+ 0 -

[Reply](#)



Chandra

Thanks John, I have been following your newsletters for a long time. This is a wonderful vlog. Looking forward to more.

+ 0 -

[Reply](#)



John Teel

Author

Thank you Chandra, I really appreciate that!

+ 0 -

[Reply](#)



Simon Robb

Thank you for all your hard work John.

Guest

I'm so impressed with your content. I come from a software background and started tinkering with hardware about 18 months ago, in order to develop precision agriculture sensors for my (aspiring) startup. It was a struggle at times – frustrating experience. It seems every paragraph of yours I read, I recognize it as a hard-won lesson from my first six months.

So, well done on so carefully curating these lessons. I would not be at all surprised if this syllabus enables a wave of makers!

+ 0 -

[Reply](#)



John Teel

Author

Thank you so much Simon for your fantastic comment. It always means so much to hear from those less experienced from all the hard work I put into creating this content.

Best wishes,

John

+ 0 -

[Reply](#)

**DApo**

Very educative

Guest

+ 0 -

Reply

**Stevie**

As a beginner to microcontrollers I am finding your articles informative...

Guest

Thanks you for helping me to understand the basics of this gigantic topic.

+ 0 -

Reply

**John Teel**

Author

That's awesome to hear Stevie! Thank you for the positive feedback, I really appreciate it.

+ 0 -

Reply

**Magnus Olsson**

Guest

I am at medium beginners leveland due to English not being my native language some (quite a lot) went head in this fast fast video. But I'm learning step by step and it's always nice to see a pro in action.

+ 0 -

Reply

**Qandeel**

Great article !

Guest

+ 0 -

Reply

**Santosh**

Thanks , I am new to this field , the way u r explaining is really very exciting.Waiting to see ur next tutorial.

Guest

+ 0 -

Reply

**John Teel**

Author

Thank you Santosh!

+ 0 -

Reply

Stefano Reksten

Thanks John! I really appreciate all the knowledge you share with us!

Guest

+ 0 -

Reply



John Teel

Author

You are welcome, and I truly do enjoy sharing it. Thanks for the comment.

John

+ 0 -

Reply

Ruben Tjade

Thank You John, it is really a pleasure again to read your article.

I have really learned so many thing.

I will like to know if all the microcontroller needs a Bootloader to operate like Arduino series?

Guest

+ 0 -

Reply



John Teel

Author

Thanks Ruben! Yes, you would need a bootloader to program it via the USB port like an Arduino. In most cases when programming a custom MCU board its best to use the in-circuit programming/debugging which for the STM32 is either SWD or JTAG. You'll need an [ST-LINK](#) programmer to program via SWD.

+ 0 -

Reply

Mes

Good one John. It is one of the few Blog/vlogs that I follow regularly. I have two simple questions if possible:
1. Is it ok to mix the two grounds supplies (Analogue and Digital) for mixed signal boards?
2. what do you recommend to use where an IC has two different VDDs like 3V and 1.8V?

Regards!

Guest

+ 0 -

Reply



John Teel

Author

Thank you and that is great to hear!

1) Yes, ideally you want to keep the digital and analog grounds separate and then connect them together to the ground plane. But for this first simple design this wasn't necessary, especially since we're not yet using an Analog-to-Digital Converter. This is something I'll discuss further in a future tutorial.

2) I'm not quite sure I understand this question. But if you use an IC that requires two supply voltages you'll need two regulators (or a single dual-output regulator) to provide those voltages. If the 1.8V supply has to only supply small amounts of current then a linear regulator is best. But if the 1.8V supply has to supply high current then a switching regulator is best. Whenever you have a situation where there is high current draw and a big difference in the input voltage to output voltage then a switching regulator will be much more efficient. So if you take a 5V supply and step it down to 1.8V and supply hundreds of millamps then a switching reg is probably the best choice.

Okay, I hope I answered your questions. Please let me know if I didn't and I'll be happy to try again:)

Best wishes,

John

+ 0 -

[Reply](#)



Rob

This is just what we needed! Looking forward to the next installment!

Guest

+ 0 -

[Reply](#)



Johan

Thanks!

Guest

+ 0 -

[Reply](#)



Jorge

Congratulations!! I think that your newsletter / mail is the only that I always read. You're not spam. that's a compliment!! Keep going!!

Guest

+ 0 -

[Reply](#)



John Teel

Author

That's awesome to hear Jorge! Thanks for the comment.

John

+ 0 - [Reply](#)



Guest

Nick

I am really loving your site John. This micro controller series is just what I needed. Thank you and I hope to see more.

-Nick.

+ 0 - [Reply](#)



Guest

Chris

The only newsletter I look forward to opening, knowing I'll get something useful out of it. Thank you for great content, John!

+ 0 - [Reply](#)



John Teel

Author

Thank you Chris that means a lot to me to hear.

John

+ 0 - [Reply](#)



Guest

Gabriel Cataldi

Really good!!!

+ 0 - [Reply](#)



John Teel

Author

Thanks Gabriel!

+ 0 - [Reply](#)



Guest

Craig Ross

Enjoyed the tutorial, looking forward to the next two installments!

+ 0 - [Reply](#)



John Teel

Thanks Craig!

Author

+ 0 -

Reply

←PREVIOUS POST

NEXT POST→

[HOME](#) [ABOUT](#) [BLOG](#) [RESOURCES](#) [CONTACT](#) [REPORT](#) [ACADEMY](#)

Predictable Designs LLC, 10645 N. Oracle Rd, Suite 121-117, Oro Valley, AZ 85737 USA

Info@PredictableDesigns.com, Phone: (520) 261-1844 (for fastest response please email instead of call)

Copyright 2019 by Predictable Designs LLC

This website uses cookies. By continuing you consent to the use of cookies and the collection of your IP address. See our [privacy policy](#) for more information.