I'm not a robot

0 ratings0% found this document useful (0 votes)351 viewsSaveSave Arduino Uno Pin Diagram, Specifications, Pin Confi... For Later0%0% found this document useful, undefined0 ratings0% found this document useful (0 votes)351 views Arduino Uno is a popular microcontroller development board based on 8-bit ATmega328P microcontroller. Along with ATmega328P MCU IC, it consists of other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. This article explores the Arduino UNO pin diagram in detail along with basics on how to use this board and upload your first code. Please note that this article discusses the popular Arduino UNO R3 development board and not the latest Arduino UNO R4 board which was launched recently. If you are new you can also check out our article on Arduino UNO R3 vs Arduino Arduino UNO R4 to understand the differences between these two boards. Arduino Uno Pinout Configuration Pin Category Pin Name Pin Description Power Vin, 3.3V, 5V, GND Vin: Input voltage to Arduino when using an external power source. 5V: Regulated power supply used to power microcontroller and other components on the board. 3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA. GND: ground pins. Reset Reset Resets the microcontroller. Analog Pins A0 – A5 Used to provide analog input in the range of 0-5V Input/Output Pins Digital Pins 0 - 13 Can be used as input or output pins. Serial 0(Rx), 1(Tx) Used to receive and transmit TTL serial data. External Interrupts 2, 3 To trigger an interrupt. PWM 3, 5, 6, 9, 11 Provides 8-bit PWM output. SPI 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK) Used for SPI communication. Inbuilt LED 13 To turn on the inbuilt LED. TWI A4 (SDA), A5 (SCA) Used for TWI communication. AREF AREF To provide reference voltage for input voltage. Arduino Uno Technical Specifications Microcontroller ATmega328P – 8 bit AVR family microcontroller Operating Voltage 5V Recommended Input Voltage 7-12V Input Voltage Limits 6-20V Analog Input Pins 6 (A0 – A5) Digital I/O Pins 14 (Out of which 6 provide PWM output) DC Current on I/O Pins 40 mA DC Current on 3.3V Pin 50 mA Flash Memory 32 KB (0.5 KB is used for Bootloader) SRAM 2 KB EEPROM 1 KB Frequency (Clock Speed) 16 MHz Note: Complete technical information can be found in the Arduino UNO Datasheet, linked at the bottom of this page. Other Arduino Boards Arduino Nano, Arduino Pro Mini, Arduino Mega, Arduino Due, Arduino MKR1000 Wi-Fi Board, Arduino Leonardo Overview Arduino Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. Arduino Uno has 14 digital input/output pins (out of which 6 can be used as PWM outputs), 6 analog input pins, a USB connection, A Power barrel jack, an ICSP header and a reset button. Arduino UNO Pin Layout Overview The Arduino UNO pin layout is organized into distinct categories, including Power Pins, Digital Pins, Analog Pins, and Special Function Pins. Each category plays a specific role in enabling the functionality of the board. The Arduino UNO pinouts available under each category is shown in the image below Now, lets understand the function of each pin under in detail under different category Power Pins Power pins are essential for operating the board and connected devices. The main pins include: VIN: Accepts external power sources (7-12V). 5V and 3.3V: Provide regulated voltage outputs for peripherals. GND (Ground): Completes the circuit. IOREF: Supplies a voltage reference for I/O pins. Tip: Always verify the voltage compatibility of connected components to avoid damage. Digital Pins (0-13) The Arduino UNO has 14 digital pins that can function as inputs or outputs. Pins 0 (RX) and 1 (TX): Reserved for serial communication. Pins 2-13: General-purpose I/O pins. PWM Pins (3, 5, 6, 9, 10, 11): Support Pulse Width Modulation, ideal for applications like controlling motors and dimming LEDs. Use functions like pinMode(), digitalWrite(), and digitalRead() to interact with these pins. Analog Pins (A0-A5) Analog pins allow reading continuous voltage signals, often from sensors. Resolution: 10-bit (0 to 1023 range). Flexibility: Can also function as digital I/O pins when required. Special Function Pins Reset Pin: Resets the board when triggered. AREF: Used to provide an external voltage reference for analog inputs. Serial Pins (RX/TX): Facilitate UART communication for serial data exchange. ICSP Header The ICSP (In-Circuit Serial Programming) header allows direct programming of the microcontroller or connecting additional peripherals. MISO (Master-In-Slave-Out): Receives data from peripherals. MOSI (Master-Out-Slave-In): Sends data to peripherals. SCK (Serial Clock): Synchronizes data transfer. Communication Pins I2C Pins: SCL (Clock line) and SDA (Data line) are located on A5 and A4, respectively. SPI Pins: Share functionality with the ICSP header (MISO, MOSI, and SCK). UART Pins: TX (Pin 1) and RX (Pin 0) handle serial communication. Arduino Uno to ATmega328 Pin Mapping When ATmega328 chip is used in place of Arduino Uno, or vice versa, the image below shows the pin mapping between the two. Software (Arduino IDE) Arduino IDE (Integrated Development Environment) is required to program the Arduino Uno board. Download it from here. Programming Arduino Once arduino IDE is installed on the computer, connect the board with computer using USB cable. Now open the arduino IDE and choose the correct board by selecting Tools>Boards>Arduino/Genuino Uno, and choose the correct Port by selecting Tools>Port. Arduino Uno is programmed using Arduino programming language based on Wiring. To get it started with Arduino Uno board and blink the built-in LED, load the example code by selecting Files>Examples>Basics>Blink. Once the example code (also shown below) is loaded into your IDE, click on the ‘upload’ button given on the top bar. Once the upload is finished, you should see the Arduino’s built-in LED blinking. Below is the example code for blinking: // the setup function runs once when you press reset or power the board void setup() { // initialize digital pin LED_BUILTIN as an output. pinMode(LED_BUILTIN, OUTPUT); } // the loop function runs over and over again forever void loop() { digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level) delay(1000); // wait for a second digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW delay(1000); // wait for a second } Applications Prototyping of Electronics Products and Systems Multiple DIY Arduino Projects. Easy to use for beginner level DIYers and makers. Projects requiring Multiple I/O interfaces and communications. Commonly Asked Question when working with Arduino UNO Q1. What is the function of PWM pins on Arduino UNO? PWM pins generate variable output signals for tasks like motor control and dimming LEDs. Q2. Can I use analog pins as digital pins? Yes, analog pins (A0-A5) can be configured as digital I/O pins. Q3. What is the role of the ICSP header? It allows direct programming of the microcontroller or connecting advanced peripherals. Q4. What happens if I supply more than 5V to an I/O pin? Excess voltage can permanently damage the microcontroller. Q5. Are all digital pins PWM-capable? No, only pins 3, 5, 6, 9, 10, and 11 support PWM. Q6. How do I power the Arduino UNO? You can use the USB port, VIN pin, or DC power jack. 2D Model and Dimensions The Arduino UNO board has over 20 pins that you can use for various applications. In this tutorial, we’ll explore the Arduino UNO Pinout and break down each section of Arduino Pins to discuss its functionality and use in detail. You’ll learn how to use Arduino Pins in your project and get a solid understanding of a lot of Arduino fundamentals.Moreover, you’ll also get the Arduino UNO pinout diagram and the Arduino R3 schematic diagram that you can use to create your own Arduino PCB board with any custom modifications that you’d need for your project. Without further ado, let’s get right into it!Table of ContentsArduino UNO Pinout GuideIn the previous tutorial (Getting Started With Arduino), we discussed a lot of Arduino Hardware/Software fundamentals and the Arduino Ecosystem, we also prepared the development/simulation/prototyping environments, and we were able to develop and test our first 3 beginner projects.Now, we’ll get a closer look at the Arduino UNO pinout and discuss the usage for each pin and its intended functionality. So, in this tutorial you’ll mainly get the following points:Arduino UNO Pinout DiagramWhat are the Arduino pins you can use?What are their functionalities in detail?Some examples & use cases for each set of Arduino pinsDespite the fact that this article’s main focus is Arduino UNO Pinout, it actually is still however very useful for Arduino beginners with different Arduino boards. The knowledge and basics we’ll be discussing here are transferable to any other Arduino board even if it’s not AVR-based.Arduino UNO Pinout DiagramHere is a detailed Arduino UNO Pinout Diagram with colored labels and legends to help you easily identify the functionalities that each pin can perform. It also has the Atmega328p AVR microcontroller pinout diagram to ease the mapping from the Arduino IO numbered pins to the actual microcontroller’s pins.You can right-click and open in a new tab to view the full-resolution image and/or download it to your computer for reference.“Pinout of ARDUINO Board and ATMega328PU” by pighixxx is licensed under Creative Commons Attribution-Share Alike 4.0 InternationalArduino UNO Pinout – Power PinsThe Arduino UNO board has a couple of LDO voltage regulators that provide regulated DC voltage of +5v and +3.3v to power up the main Arduino (Atmega328p) microcontroller. And you can also use the power pins to power up external modules and sensors that you’d like to use with your Arduino board.There are multiple options when it comes to powering up your Arduino UNO board that we’ll discuss in detail hereafter in this section. Here is a figure for the power section on your Arduino UNO board with a detailed pinout for each power pin and its functionality.Arduino Ground PinsThe Arduino UNO board has 3 ground pins that you can use while connecting external devices like LEDs, Sensors, Modules, etc.+5v SupplyThis is a regulated DC voltage output of +5v that you can use to supply external sensors and modules from your Arduino board. Try not to overload it with heavy loads.+3.3v SupplyThis is a regulated DC voltage output of +3.3v that you can use to supply external sensors and modules from your Arduino board. As some sensors and modules are not 5v tolerant, you need to supply them with 3.3v DC and also use a level-shifter with Arduino IO pins as they are also at 5v level.Arduino VIN PinFor powering your Arduino UNO board, you have a few options to do so. The first of which is to use the USB cable to provide power to your Arduino board and be able to flash a new firmware to the microcontroller over the USB communication.Another option is to use the DC jack input to supply the Arduino board with DC voltage (6v-12v) and try not to exceed 9v because higher voltage input will cause the LDO regulators to overheat and potentially get damaged quickly.The last option is to use the VIN pin to provide an external voltage supply that’s also within the (6v-12v) range exactly like the DC-Jack input. And do not under any circumstances connect multiple power sources at the same time. Use only one option of the 3 we’ve discussed before.Arduino IOREF PinThe Arduino IOREF pin provides a 5v logic voltage reference for Arduino Shields that need to use it.Arduino UNO Pinout – Digital PinsArduino GPIO (General-Purpose Input Output) Pins are the digital IO pins of the Arduino UNO board.There are 14 GPIO pins on the Arduino UNO board as shown in the figure below.IO PinsThe pins (0 to 13) are used as a digital input or output for the Arduino UNO board.Pin 13 is used for the onboard (built-in) LED.In output mode, the Arduino digital pin is:5v (when the digital pin is set HIGH)0v (when the digital pin is set to LOW).In input mode, the Arduino digital input pin will read a 0 or 1 depending on the voltage applied to the pin:0v–0.8v: read as 0 (logic 0 or LOW)2v–5v: read as 1 (logic 1 or HIGH)Any voltage level between 0.8v-2v is considered undefined input for the digital IO pin.There are two protection diodes for each IO pin that will clip excessive over/under voltage away from damaging any IO pin. However, you still need to be careful with the voltage levels of various components in your system and do level-shifting if needed.PWM PinsThe pins indicated by the “~” on the Arduino board are the PWM output pins. There are 6 PWM output pins on the Arduino UNO board (pins 3, 5, 6, 9, 10, and 11).The PWM (Pulse Width Modulation) is a timer mode in which the timer is used to generate a constantly changing digital signal with a specific frequency and duty cycle. PWM is widely used for so many applications like motor control, LED brightness control, and much more.If you need more than 6 PWM outputs for your application, you can either check another board other than Arduino UNO or search for an external PWM controller IC to extend the PWM output pins count. There is also a possibility to create Software PWM (software emulated) output signals but be careful that it’s going to cost you a lot of CPU load.Check the tutorial below to learn more about Arduino PWM.Interrupt PinsInterrupt pins are basically digital IO pins that can be configured to be in input mode. And will provide an interrupt signal directly to the CPU whenever the pin state changes. The CPU will pause the current operation and jump to a predefined interrupt handler function and execute it till completion, then it resumes the main loop() function (application).Interrupt pins can provide a near-immediate response to external events like input pin state change and this can be really useful for different applications that include, but are not limited to, the following list:Motor Speed Measurement With Optical EncoderProximity Sensor InterfacingKeypad Matrix ScanningUltrasonic Sensor Interfacingand much moreThere are two types of interrupt pins in the Arduino (Atmega328p) microcontroller: dedicated external interrupt pins and pin-change interrupts.1. External InterruptsThose are two dedicated pins INT0 and INT1 that correspond to Arduino IO pins 2, and 3 respectively. Each interrupt pin of those two has its own interrupt vector and can have its unique interrupt handler function.Note that other Arduino boards may have more dedicated external interrupt pins. The Arduino UNO microcontroller has only 2 dedicated external interrupt pins. And there is no practical way of increasing the dedicated interrupt pins.2. PCINT (Pin Change Interrupts)All Arduino UNO digital pins can be used as interrupt pins using the PCINT hardware. This is not a dedicated interrupt signal source unlike the previously discussed external interrupt pints (INT0/INT1).You can easily notice that every IO pin in the Arduino UNO pinout diagram has a PCINTx label. Where x is the number of the pin change interrupt signal. Those signals are grouped together to generate 3 interrupt signals one for each port of pins (combined together). And the only way to know which pins have their state changed is to manually check for it in your code logic.Having to manually check which pin state has changed in each port is the major drawback for PCINT interrupts compared to the dedicated external interrupts (INT0/INT1).Timer/Counter PinThere are two pins (T0 and T1) that can be used as external clock sources for Timer0 and Timer1 respectively. This can be very useful in a lot of applications depending on the mode of operation for the Timer module. (T0 is Arduino Pin4 and T1 is Arduino Pin5)In Counter Mode, the T0/T1 pins can be used as an external clock source so you can feed your counter module with external pulses so it keeps counting those incoming pulses. There are several use cases to take advantage of this hardware feature such as:Building a Frequency Counter Using Timer Module in Counter ModeMeasuring Motor Speed Using Optical Encoder + Timer in Counter Modeand much moreBy counting the incoming pulses count over a fixed time interval, we can determine the frequency (or period) of the input signal. As stated before, this can be very useful for reading sensors like optical encoders, proximity sensors for object counting, and much more.In Timer Mode, the T0/T1 pins can be used as an external clock source so you can feed your timer module with an external time-base clock generated from any other device (or circuitry) outside of the Arduino (Atmega328) microcontroller itself. One application that you can think of for this hardware feature is the RTC clock (Real-Time Clock).External RTC crystal (32.768kHz) can feed its output waveform to the T0 or T1 pin which is going to be selected as a clock source. Set the prescaler to 32. And given that Timer0/Timer1 are both 10 bits. Now you’ll get a perfect 1-second timer interrupt that you can use as a time base for your real-time clock application. TOUT = 2n x (Prescaler/FCLK), which turns out to be TOUT = 210 x (32/32768) = 1 second.ICP (Input Capture) PinsThere is one input capture pin in Arduino UNO (Atmega328p) which is ICP1 that corresponds to Arduino pin 8. This pin is used with the timer module in timer mode while it’s free running. When the ICP pin state changes, the current value of the timer register is captured and saved in the input capture register. It’s also referred to as ICU (Input Capture Unit).This can be very useful in so many applications that include the following:External input signal pulse width measurementMotor RPM Measurement With Optical Encoder + ICUUltrasonic Distance Sensor Interfacing With ICUFrequency Counter ApplicationsPOV LED Strip Displayand much moreICSP Port PinsThe ICSP (In-Circuit Serial Programming) Header provides an extra way to flash (program) the Atmega328p microcontroller on the Arduino UNO board over the SPI. However, you may rarely need to use it as long as the USB port is functioning properly.The Arduino bootloader is flashed only once to the microcontroller’s IC and then USB programming is the way to go thereafter. Unless something wrong goes with your chip or if you’ve replaced the Atmega328p microcontroller chip with a brand new one that doesn’t have the bootloader flashed to it. Then the ICSP port will come to the rescue.Arduino UNO Pinout – Analog PinsThe Arduino UNO has 6 analog input pins labeled from A0 to A5 as shown in the figure below. Those pins can be used with analog peripherals in the Arduino microcontroller such as: ADC (A/D Converter) and the Analog Comparator.The Analog input pins can also be used as general-purpose digital IO pins as we’ll see hereafter in this section.ADC (A/D Converter)The ADC (Analog to Digital Converter) in Arduino is used to read analog voltage input and convert it to its digital representation. Given that the Arduino’s ADC is 10 bits in resolution, the digital output range is therefore from 0 up to 1023. And the analog input range is from 0v up to 5v (which is the analog reference voltage VREF = +5v).You can use the interactive tool below to set an analog input voltage and see the ADC digital output value that corresponds to the analog input voltage. The output equation for the ADC is as follows: ADC Output = (Analog input voltage / VREF) x (2n – 1). Where VREF = 5v and n is the ADC resolution which is 10bits. Analog ComparatorThe analog input pins can be used with the internal Analog comparator in the Arduino’s microcontroller (Atmega328p) instead of the ADC. The analog comparator is really an interesting peripheral and can be a very powerful alternative for the ADC in many applications.Imagine you need to monitor some input analog voltage and check if it goes below a certain threshold value. You can use the ADC and keep sampling the input pin thousands of times per second and overload the CPU with this hectic task. While you could have used the analog comparator instead and set a threshold to the input voltage, when a voltage drop occurs, the analog comparator will fire an interrupt and the CPU will immediately respond to it. Without overloading the CPU with ADC conversions and polling for the analog input voltage.There are so many applications where you might need to use the analog comparator which include, but are not limited to, the following:Threshold voltage monitoringBattery Charging ApplicationsZCD (Zero-Crossing Detection)Tachometersand much moreWe’ll dedicate some future tutorials to discuss Arduino’s internal analog comparator in more detail with example projects as well.AREF Pin (Analog Reference)The AREF (Analog Reference) pin can be used to provide an external reference voltage for the analog-to-digital conversion of inputs to the analog pins (A0-A5). The reference voltage essentially specifies the maximum analog input voltage after which the ADC’s output will saturate at 1023.Let’s recall the ADC’s output equation which is: ADC Output = (Analog input voltage / VREF) x (2n – 1). Where the “default” VREF = 5v and n is the ADC resolution which is 10bits.Using the AREF pin we can change the default VREF voltage and set it to any desired voltage level which allows for more precise A/D conversions. Let’s say we hook up the AREF pin to an external 2.5v DC supply, now we’ll end up having double the resolution of the default 5v analog reference.Because instead of dividing the 0v-5v range into discrete 1024 digital levels, we’ll now divide the 0v-2.5v range into discrete 1024 digital levels. Which is exactly double the default resolution. But you must make sure that the input analog voltage will not exceed 2.5v after setting the AREF to 2.5v. Otherwise, the ADC will saturate at 1023 if the analog input voltage exceeded the new AREF (2.5v). Use the interactive tool above to try it yourself and see the effect of lowering the AREF voltage from 5v down to 2.5v and how it increases the ADC resolution by a factor of 2.VREFAnalog Input VoltageADC Digital Output5v0.01v22.5v0.01v4For the exact same analog input (0.01v), you get more ADC ticks (levels) at VREF=2.5v compared to VREF=5v. Which is what we mean by the resolution is doubled (increased by a factor of 2).Using Analog Pins As Digital IOArduino Uno has a total of 14 digital GPIO pins, out of which 6 pins (from A0 to A5) are analog pins. The question is can we use Arduino analog pins as digital output pins? The answer is YES, and here is how to do it.First of all, the ADC is a valuable resource in any microcontroller. Under certain circumstances, you’d sacrifice the analog input channels just to get extra digital IO pins. Sometimes an external IO expander would be a better solution instead of sacrificing the ADC analog input channels.But let’s say you’ve made your decision and you need to have an extra digital pin or two. And the analog input pins are the only hope you’ve got. Here is how you’d go about implementing this,Just deal with the analog input pins (A0 – A5) as if they’re normal digital IO pins. This means you first need to call the pinMode() function to set the pin mode to OUTPUT. And then call the digitalWrite() function to change the pin state to HIGH or LOW. Check out the tutorial below to get more information about this with a couple of code examples.Arduino UNO Pinout – Serial Ports PinsThe Arduino UNO’s microcontroller (Atmega328p) has 3 different serial communication ports (protocols). Those ports enable the Arduino to communicate with a huge variety of sensors and modules to extend its functionality and capabilities beyond the limits of the Atmega microcontroller.Serial communication is a method of data exchange between embedded controllers in the form of bits that are sent/received bit-by-bit in a series. Hence the name, serial communication. There are different hardware ports (protocols) for serial data communication of which (UART, I2C, and SPI) that are supported by the Arduino’s microcontroller.Each serial communication port (protocol) has its own set of advantages/disadvantages and its best use cases. We’ll dedicate a separate in-depth tutorial for each serial communication protocol of them. But here is a summarized overview of each port and its IO pins.Arduino UART PinsUART (Universal Asynchronous Receiver-Transmitter) is the most popular serial communication protocol in embedded microcontrollers. In Arduino, we typically use the UART module for serial communication with the PC via a USB-TTL converter to print serial messages on the serial monitor.The Arduino UNO has only one UART module which has two pins (RX for receiving serial data and TX for transmitting serial data). The UART pins’ mapping is as follows: RX is Arduino pin 0, and TX is Arduino pin 1 respectively.Besides serial communication with your PC for debugging and such, the UART is also used for interfacing various sensors and modules such as:Bluetooth Module HC-05GSM/GPRS ModulesGPS Modulesand much moreArduino I2C PinsI2C or IIC (Inter-Integrated Circuit) is also a very popular serial communication protocol that’s widely used by different sensors and modules in embedded systems. It consists of 2 pins only (one for serial data and one for the serial clock). The Arduino I2C pins are (SDA and SCL).The I2C is a multi-master multi-slave protocol that supports a large number of devices on the same 2-wire bus. Each device has a unique ID that others can use to directly address that specific device. And it’s considered the most efficient serial communication bus in terms of the number of IO pins needed to establish a communication network of multiple devices.There are so many modules and sensors that we can interface with Arduino using the I2C bus, which include:RTC ClockOLED Display ScreensI2C IO Pins ExpanderExternal EEPROM Memory ChipsMPU6050 IMU (Gyro+Accelerometer)and much moreTypically, you won’t need more than one I2C module on your Arduino microcontroller. Because if you need to add a new extra device to the I2C bus, it’s as simple as hooking it up to the SCL and SDA lines and you’re good to go. Unlike the UART which is a single master single slave communication port.Arduino SPI PinsSPI (Serial Peripheral Interface) is one of the fastest serial communication ports at all. You can use Arduino’s SPI for communicating with various sensors and modules at much higher data rates compared to the other serial communication protocols.The SPI communication bus consists of 3 main lines (Serial Data Out, Serial Data In, and Serial Clock). The SPI master device needs also an extra IO pin to select the slave to which it’d like to communicate. This means we need an extra IO pin for each slave device added to the bus which is definitely going to limit the maximum addressable devices on the bus.The SPI bus lines are named according to a conversion that eases the wiring and usage for users, which goes as follows:MOSI (Master Output Slave Input)MISO (Master Input Slave Output)SCK (Serial Clock)SS (Slave Select) “Only Required For SPI Slave Devices“There are so many applications in which you’d need to use the Arduino SPI bus to interface various sensors and modules which include, but are not limited to, the following examples:SPI TFT Display InterfacingDot Matrix Display InterfacingExternal SPI Flash MemoryExternal SPI ADC ModulesnRF24L01 Wireless Modulesand much moreAs stated earlier, We’ll dedicate a separate in-depth tutorial for each serial communication protocol (UART – I2C – SPI). And we’ll create a handful of applications and example projects to practice what we’ll learn.Those are summarized tables for the Arduino UNO pinout with brief descriptions of each pin’s functionalities.Arduino Digital IO PinsArduino PinAtmega328p PinPWM?Alternative Special FunctionDescriptionD0PD0RXUART RX PinD1PD1TXUART TX PinD2PD2INT0External Interrupt Pin0D3PD3✓INT1External Interrupt Pin1D4PD4T0Timer0 External Clock Input PinD5PD5✓T1Timer1 External Clock Input PinD6PD6✓D7PD7D8PB0ICP1Input Capture Unit PinD9PB1✓D10PB2✓SSSPI Slave Select PinD11PB3✓MOSISPI Master-Out Slave-InD12PB4MISOSPI Master-In Slave-OutD13PB5SCKSPI Serial Clock PinArduino Analog Input PinsArduino PinAtmega328p PinAlternative Special FunctionDescriptionA0PC0A1PC1A2PC2A3PC3A4PC4SDAI2C Serial Data LineA5PC5SCLI2C Serial Clock LineAREFAREFAREFAnalog Voltage Reference Input PinArduino Power PinsPin NameDescription5v+5v DC Regulated Supply3.3v+3.3v DC Regulated SupplyGNDGroundRESETµC Reset LineVINInput DC Voltage (6v-12v). Try not to exceed 9vIOREFProvides 5v Logic Voltage Reference For Arduino Shields That Use itArduino UNO R3 Schematic DiagramAs an open-source project, the Arduino UNO board schematic design and parts list is completely open-source and available for everyone to download, modify, and create custom Arduino board designs based on it.You’ll also get a much clear view of the underlying hardware details by carefully studying the Arduino UNO board schematic design files. You can download the Arduino UNO R3 schematic design diagram file using (this link).Get Started With Arduino UNO PinsTo conclude this tutorial, we’ll highlight the fact that there are so many options for using Arduino pins and a unique set of features that each pin does support. We hope it’s been a helpful guide to helping you decide which Arduino pins to use for your project and what are the fundamental limitations and workarounds for each.If you’re just getting started with Arduino, you need to check out the Arduino Getting Started [Ultimate Guide] here.And follow this Arduino Series of Tutorials to learn more about Arduino Programming. Create stunning LED animations with Arduino UNO R4 WiFi (by Kamitronix). Design, download and watch your custom animation come to life.DISCOVER MORE Home / Hardware / UNO R3 Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.The ATmega328P can easily be replaced, as it is not soldered to the board.The ATmega328P also features 1kb of EEPROM, a memory which is not erased when powered off.The Arduino UNO features a barrel plug connector, that works great with a standard 9V battery. In our last two posts, we focused on the software aspects of the Arduino. We saw that Arduino boards are programmed using a language derived from C and C++ in Arduino's Integrated Development Environment (IDE) and learned a few basic debugging methods. In this post, we'll be taking a closer look at the Arduino hardware, and more specifically, the Arduino Uno pinout. Arduino Uno is based on the ATmega328 by Atmel. The Arduino Uno pinout consists of 14 digital pins, 6 analog inputs, a power jack, USB connection and ICSP header. The versatility of the pinout provides many different options such as driving motors, LEDs, reading sensors and more. In this post, we’ll go over the capabilities of the Arduino Uno pinout.Start Your Arduino Circuit Arduino Uno Pinout - Diagram“Pinout of ARDUINO Board and ATMega328PU” by pighixxx is licensed under Creative Commons Attribution-Share Alike 4.0 InternationalArduino Uno pinout - Power SupplyThere are 3 ways to power the Arduino Uno:Barrel Jack - The Barrel jack, or DC Power Jack can be used to power your Arduino board. The barrel jack is usually connected to a wall adapter. The board can be powered by 5-20 volts but the manufacturer recommends to keep it between 7-12 volts. Above 12 volts, the regulators might overheat, and below 7 volts, might not suffice.VIN Pin - This pin is used to power the Arduino Uno board using an external power source. The voltage should be within the range mentioned above.USB cable - when connected to the computer, provides 5 volts at 500mA.There is a polarity protection diode connecting between the positive of the barrel jack to the VIN pin, rated at 1 Ampere.The power source you use determines the power you have available for your circuit. For instance, powering the circuit using the USB limits you to 500mA. Take into consideration that this is also used for powering the MCU, its peripherals, the on-board regulators, and the components connected to it. When powering your circuit through the barrel jack or VIN, the maximum capacity available is determined by the 5 and 3.3 volts regulators on-board the Arduino.They provide regulated 5 and 3.3v to power external components according to manufacturer specifications.In the Arduino Uno pinout, you can find 5 GND pins, which are all interconnected.The GND pins are used to close the electrical circuit and provide a common logic reference level throughout your circuit. Always make sure that all GNDs (of the Arduino, peripherals and components) are connected to one another and have a common ground.RESET - resets the ArduinoIOREF - This pin is the input/output reference. It provides the voltage reference with which the microcontroller operates.Arduino Uno Pinout - Analog INThe Arduino Uno has 6 analog pins, which utilize ADC (Analog to Digital converter).These pins serve as analog inputs but can also function as digital inputs or digital outputs.Analog to Digital ConversionADC stands for Analog to Digital Converter. ADC is an electronic circuit used to convert analog signals into digital signals. This digital representation of analog signals allows the processor (which is a digital device) to measure the analog signal and use it through its operation.Arduino Pins A0-A5 are capable of reading analog voltages. On Arduino the ADC has 10-bit resolution, meaning it can represent analog voltage by 1,024 digital levels. The ADC converts voltage into bits which the microprocessor can understand.One common example of an ADC is Voice over IP (VoIP). Every smartphone has a microphone that converts sound waves (voice) into analog voltage. This goes through the device’s ADC, gets converted into digital data, which is transmitted to the receiving side over the internet.Arduino Uno Pinout - Digital PinsPins 0-13 of the Arduino Uno serve as digital input/output pins.Pin 13 of the Arduino Uno is connected to the built-in LED.In the Arduino Uno - pins 3,5,6,9,10,11 have PWM capability.It’s important to note that:● Each pin can provide/sink up to 40 mA max. But the recommended current is 20 mA.● The absolute max current provided (or sank) from all pins together is 200mAWhat does digital mean?Digital is a way of representing voltage in 1 bit: either 0 or 1. Digital pins on the Arduino are pins designed to be configured as inputs or outputs according to the needs of the user. Digital pins are either on or off. When ON they are in a HIGH voltage state of 5V and when OFF they are in a LOW voltage state of 0V.On the Arduino, When the digital pins are configured as output, they are set to 0 or 5 volts.When the digital pins are configured as input, the voltage is supplied from an external device. This voltage can vary between 0-5 volts which is converted into digital representation (0 or 1). To determine this, there are 2 thresholds:● Below 0.8v - considered as 0.● Above 2v - considered as 1.When connecting a component to a digital pin, make sure that the logic levels match. If the voltage is in between the thresholds, the returning value will be undefined.What is PWM?In general, Pulse Width Modulation (PWM) is a modulation technique used to encode a message into a pulsing signal. A PWM is comprised of two key components: frequency and duty cycle. The PWM frequency dictates how long it takes to complete a single cycle (period) and how quickly the signal fluctuates from high to low. The duty cycle determines how long a signal stays high out of the total period. Duty cycle is represented in percentage.In Arduino, the PWM enabled pins produce a constant frequency of ~ 500Hz, while the duty cycle changes according to the parameters set by the user. See the following illustration:PWM signals are used for speed control of DC motors, dimming LEDs and more.Communication ProtocolsSerial (TTL) - Digital pins 0 and 1 are the serial pins of the Arduino Uno.They are used by the onboard USB module.What is Serial Communication? Serial communication is used to exchange data between the Arduino board and another serial device such as computers, displays, sensors and more. Each Arduino board has at least one serial port. Serial communication occurs on digital pins 0 (RX) and 1 (TX) as well as via USB. Arduino supports serial communication through digital pins with the SoftwareSerial Library as well. This allows the user to connect multiple serial-enabled devices and leave the main serial port available for the USB.Software serial and hardware serial - Most microcontrollers have hardware designed to communicate with other serial devices. Software serial ports use a pin-change interrupt system to communicate. There is a built-in library for Software Serial communication. Software serial is used by the processor to simulate extra serial ports. The only drawback with software serial is that it requires more processing and cannot support the same high speeds as hardware serial.SPI - SS/SCK/MISO/MOSI pins are the dedicated pins for SPI communication. They can be found on digital pins 10-13 of the Arduino Uno and on the ICSP headers.What is SPI?Serial Peripheral Interface (SPI) is a serial data protocol used by microcontrollers to communicate with one or more external devices in a bus like connection. The SPI can also be used to connect 2 microcontrollers. On the SPI bus, there is always one device that is denoted as a Master device and all the rest as Slaves. In most cases, the microcontroller is the Master device. The SS (Slave Select) pin determines which device the Master is currently communicating with.SPI enabled devices always have the following pins:MISO (Master In Slave Out) - A line for sending data to the Master deviceMOSI (Master Out Slave In) - The Master line for sending data to peripheral devicesSCK (Serial Clock) - A clock signal generated by the Master device to synchronize data transmission.I2C - SCL/SDA pins are the dedicated pins for I2C communication. On the Arduino Uno they are found on Analog pins A4 and A5.What is I2C? I2C is a communication protocol commonly referred to as the “I2C bus”. The I2C protocol was designed to enable communication between components on a single circuit board. With I2C there are 2 wires referred to as SCL and SDA.SCL is the clock line which is designed to synchronize data transfers.SDA is the line used to transmit data.Each device on the I2C bus has a unique address, up to 255 devices can be connected on the same bus.Aref - Reference voltage for the analog inputs.Interrupt - INT0 and INT1. Arduino Uno has two external interrupt pins.External Interrupt - An external interrupt is a system interrupt that occurs when outside interference is present. Interference can come from the user or other hardware devices in the network. Common uses for these interrupts in Arduino are reading the frequency a square wave generated by encoders or waking up the processor upon an external event.Arduino has two forms of interrupt:There are two external interrupt pins on the ATmega168/328 called INT0 and INT1. both INT0 and INT1 are mapped to pins 2 and 3. In contrast, Pin Change interrupts can be activated on any of the pins.Arduino Uno Pinout - ICSP HeaderICSP stands for In-Circuit Serial Programming. The name originated from In-System Programming headers (ISP). Manufacturers like Atmel who work with Arduino have developed their own in-circuit serial programming headers. These pins enable the user to program the Arduino boards’ firmware. There are six ICSP pins available on the Arduino board that can be hooked to a programmer device via a programming cable.Know your PinoutThe Arduino Uno Microcontroller is one of the most versatile boards on the market today and that’s why we decided to focus on it in this guide. This guide displays most of its capabilities, but there are also more advanced options which we did not go into in this post.The important thing to know when you choose a board for your project is its capabilities and limitations. It’s also important to understand the different communication protocols that the board uses. Of course, you don’t need to remember all of this information, you can always go back to this post and read the relevant information for you (this is a good time to bookmark this post btw).Start Your Arduino Circuit If you have any comments or questions, you are welcome to write them below and of course, feel free to share this post with your Arduino-loving friends =D In this tutorial, we will have a brief overview about the Arduino UNO Board, the layout of a typical UNO board, some of the important specifications and finally the Arduino UNO Pinout along with the pin description. Arduino UNO Pinout Perhaps the most popular board in the Arduino line-up is the Arduino UNO. There are other boards like the Arduino Nano and the Arduino Mega, but UNO has been the go-to board for quick prototyping, Arduino Projects and DIY Projects. Arduino UNO in DIP and SMD Packages Arduino UNO is based on ATmega328P Microcontroller, an 8-bit AVR Architecture based MCU from ATMEL. Arduino UNO comes in two variants: one consists of a 28-pin DIP Microcontroller while the other consists of 32 lead Quad Flat Package Microcontroller. Other than that, rest is identical in both the boards. Arduino UNO Board Layout The following image shows the layout of a typical Arduino UNO board. All the components are placed on the top side of the PCB. Layout of Arduino UNO Board As you can notice, there is a Type-B USB connector on the left short edge of the board, which is used for powering on the board as well as programming the Microcontroller. There is also a 2.1 mm DC jack to provide external power supply. Apart from that, the layout of Arduino UNO is very much self-explanatory. I will discuss about the pins of Arduino UNO in the Arduino UNO Pinout Section. Technical Specifications of Arduino UNO As Arduino UNO is based on ATmega328P Microcontroller, the technical specifications of Arduino UNO are mostly related to the ATmega328P MCU. But none the less, let me give you a brief overview about some important specifications of Arduino UNO. MCU ATmega328P Architecture AVR Operating Voltage 5V Input Voltage 6V – 20V (limit) 7V – 12V (recommended) Clock Speed 16 MHz Flash Memory 32 KB (2 KB of this used by bootloader) SRAM 2 KB EEPROM 1 KB Digital IO Pins 24 (of which 6 can produce PWM) Analog Input Pins 6 How to power up the Arduino UNO? There are a couple of ways in which you can power the UNO board. The first and easy way is using the Type-B USB Connector. The next way is to provide an unregulated supply in the range of 6V to 20V to VIN pin of the UNO (Pin number 26). You can also supply the unregulated supply through the 2.1mm DC Jack, in which case, you can access the supplied voltage through the VIN Pin. What are Different Memories of Arduino UNO? Strictly speaking, this is specific to the MCU i.e., ATmega328P, used on the Arduino UNO Board. ATmega328P microcontroller comes with three different types of memories. They are: 32 KB of Flash Memory 2 KB of SRAM 1 KB of EEPROM 0.5 KB of the Flash Memory is used by the bootloader code. What are the Input and Output Pins of Arduino UNO? Of the 32 pins available on the UNO board, 22 pins are associated with input and output. In that 14 pins (D0 to D13) are true digital IO pins, which can be configured as per you application using pinMode(), digitalWrite() and digitalRead() functions. All these Digital IO pins are capable of sourcing or sinking 20mA of current (maximum 40m

Cortex-M0 / Cortex-M0+ / Cortex-M1 can execute without modification on the Cortex-M3 / Cortex-M4 / Cortex-M7. Binary instructions available for the Cortex-M3 can execute without modification on the Cortex-M4 / Cortex-M7 / Cortex-M33 / Cortex-M35P.[15][16] Only Thumb-1 and Thumb-2 instruction sets are supported in Cortex-M architectures; the legacy 32-bit ARM instruction set isn't supported. All Cortex-M cores implement a common subset of instructions that consists of most Thumb-1, some Thumb-2, including a 32-bit result multiply. The Cortex-M0 / Cortex-M0+ / Cortex-M1 / Cortex-M23 were designed to create the smallest silicon die, thus having the fewest instructions of the Cortex-M family. The Cortex-M0 / M0+ / M1 include Thumb-1 instructions, except new instructions (CBZ, CBNZ, IT) which were added in ARMv7-M architecture. The Cortex-M0 / M0+ / M1 include a minor subset of Thumb-2 instructions (BL, DMB, DSB, ISB, MRS, MSR).[15] The Cortex-M3 / M4 / M7 / M33 / M35P have all base Thumb-1 and Thumb-2 instructions. The Cortex-M3 adds three Thumb-1 instructions, all Thumb-2 instructions, hardware integer divide, and saturation arithmetic instructions. The Cortex-M4 adds DSP instructions and an optional single-precision floating-point unit (VFPv4-SP). The Cortex-M7 adds an optional double-precision FPU (VFPv5).[23][16] The Cortex-M23 / M33 / M35P / M52 / M55 / M85 add TrustZone instructions. ARM Cortex-M instruction variations Arm Core CortexM0[18] CortexM0+[19] CortexM1[20] CortexM3[21] CortexM4[22] CortexM7[23] CortexM23[24] CortexM33[25] CortexM35P CortexM52[26] CortexM55[27] CortexM85[28] ARM architecture ARMv6-M[15] ARMv6-M[15] ARMv6-M[15] ARMv7-M[16] ARMv7E-M[16] ARMv7E-M[16] ARMv8-MBaseline[31] ARMv8-MMainline[31] ARMv8-MMainline[31] Armv8.1-MMainline[31] Armv8.1-MMainline[31] Armv8.1-MMainline[31] Computer architecture VonNeumann VonNeumann VonNeumann Harvard Harvard Harvard VonNeumann Harvard Harvard Harvard Harvard Harvard Instruction pipeline 3 stages 2 stages 3 stages 3 stages 3 stages 6 stages 2 stages 3 stages 3 stages 4 stages 4-5 stages 7 stages Interrupt latency(zero wait state memory) 16 cycles 15 cycles 23 for NMI,26 for IRQ 12 cycles 12 cycles 12 cycles,14 worstcase 15 cycles,24 secureto NS IRQ 12 cycles,21 secureto NS IRQ TBD TBD TBD TBD Thumb-1 instructions Most Most Most Most Entire Entire Entire Most Entire Entire Entire Entire Entire Thumb-2 instructions Some Some Some Entire Entire Entire Some Entire Entire Entire Entire Entire Multiply instructions32×32 = 32-bit result Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Multiply instructions32×32 = 64-bit result No No No Yes Yes Yes Yes Yes Yes Yes Yes Yes Divide instructions32/32 = 32-bit quotient No No No Yes Yes Yes Yes Yes Yes Yes Yes Yes Saturated math instructions No No No Some Yes Yes Yes Yes Yes Yes DSP instructions No No No No No Yes Yes No Optional Optional Yes Yes Yes Half-Precision (HP)floating-point instructions No No No No No No No No Optional Optional Optional Single-Precision (SP)floating-point instructions No No No No Optional Optional No Optional Optional Optional Optional Optional Optional Double-Precision (DP)floating-point instructions No No No No No Optional No No Optional Optional Optional Optional Helium vector instructions No No No No No No No No No Optional Optional Optional TrustZone security instructions No No No No No Optional Optional Optional Optional Optional Yes Co-processor instructions No No No No No No No Optional Optional Optional Optional Optional ARM Custom Instructions (ACI) No No No No No No No Optional No Optional Optional Optional Pointer Authentication and Branch TargetIdentification (PACBTI) instructions No No No No No No No No No Optional No Optional Note: Interrupt latency cycle count assumes: 1) stack located in zero-wait state RAM, 2) another interrupt function not currently executing, 3) Security Extension option doesn't exist, because it adds additional cycles. The Cortex-M cores with a Harvard computer architecture have a shorter interrupt latency than Cortex-M cores with a Von Neumann computer architecture. Note: The Cortex-M series includes three new 16-bit Thumb-1 instructions for sleep mode: SEV, WFE, WFI. Note: The Cortex-M0 / M0+ / M1 doesn't include these 16-bit Thumb-1 instructions: CBZ, CBNZ, IT.[15][16] Note: The Cortex-M0 / M0+ / M1 only include these 32-bit Thumb-2 instructions: BL, DMB, DSB, ISB, MRS, MSR.[15][16] Note: The Cortex-M0 / M0+ / M1 / M23 only has 32-bit multiply instructions with a lower-32-bit result (32 bit × 32 bit = lower 32 bit), where as the Cortex-M3 / M4 / M7 / M33 / M35P includes additional 32-bit multiply instructions with 64-bit results (32 bit × 32 bit = 64 bit). The Cortex-M4 / M7 (optionally M33 / M35P) include DSP instructions for (16 bit × 16 bit = 32 bit), (32 bit × 16 bit = upper 32 bit), (32 bit × 32 bit = upper 32 bit) multiplications.[15][16] Note: The number of cycles to complete multiply and divide instructions vary across ARM Cortex-M core designs. Some cores have a silicon option for the choice of fast speed or small size (slow speed), so cores have the option of using less silicon with the downside of higher cycle count. An interrupt occurring during the execution of a divide instruction or slow-iterative multiply instruction will cause the processor to abandon the instruction, then restart it after the interrupt returns. Multiply instructions "32-bit result" – Cortex-M0/M0+/M23 is 1 or 32 cycle silicon option, Cortex-M1 is 3 or 33 cycle silicon option, Cortex-M3/M4/M7/M33/M35P is 1 cycle. Multiply instructions "64-bit result" – Cortex-M3 is 3–5 cycles (depending on values), Cortex-M4/M7/M33/M35P is 1 cycle. Divide instructions – Cortex-M3/M4 is 2–12 cycles (depending on values), Cortex-M7 is 3–20 cycles (depending on values), Cortex-M23 is 17 or 34 cycle option, Cortex-M33 is 2–11 cycles (depending on values), Cortex-M35P is TBD. Note: Some Cortex-M cores have silicon options for various types of floating point units (FPU). The Cortex-M55 / M85 has an option for half-precision (HP), the Cortex-M4 / M7 / M33 / M35P / M52 / M55 / M85 has an option for single-precision (SP), the Cortex-M7 / M52 / M55 / M85 has an option for double-precision (DP). When an FPU is included, the core is sometimes referred as "Cortex-MxF", where 'x' is the core variant, such as Cortex-M4F.[15][16] ARM Cortex-M instruction groups Group Instrbits Instructions CortexM0, M0+, M1 CortexM3 CortexM4 CortexM7 CortexM23 CortexM33 CortexM35P CortexM52 CortexM55 CortexM85 Thumb-1 16 ADC, ADD, ADR, AND, ASR, B, BIC, BKPT, BLX, BX, CMN, CMP, CPS, EOR, LDM, LDR, LDRB, LDRH, LDRSB, LDRSH, LSL, LSR, MOV, MUL, MVN, NOP, ORR, POP, PUSH, REV, REV16, REVSH, ROR, RSB, SBC, SEV, STM, STR, STRB, STRH, SUB, SVC, SXTB, SXTH, TST, UXTB, UXTH, WFE, WFI, YIELD Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes