

1. Overview	3
1.1. Supported operating systems	6
2. Contacting Pololu	7
3. A-Star 32U4 Robot Controller with Raspberry Pi Bridge	8
3.1. Microcontroller	8
3.2. User interface	8
3.3. LV motor drivers	10
3.4. SV motor drivers	12
3.5. Power	14
3.6. LV regulator	16
3.7. SV regulator	18
3.8. Raspberry Pi interface and level shifters	19
3.9. Pinout	22
3.10. Pin assignments	25
3.11. AVR Timers	28
3.12. Schematics and dimensions	28
4. Getting started	30
4.1. Installing Windows drivers	30
4.2. Programming using the Arduino IDE	32
4.3. Programming using avr-gcc and AVRDUDE	37
5. A-Star 32U4 Arduino library	40
6. The A-Star 32U4 USB interface	41
7. The A-Star 32U4 Bootloader	43
8. Reviving an unresponsive A-Star	46
8.1. Reviving using the Arduino IDE	46
8.2. Reviving using AVRDUDE	48
9. Related resources	50

1. Overview

The Pololu A-Star 32U4 Robot Controller with Raspberry Pi Bridge is a programmable module designed to be the core of a small robot, either as an auxiliary controller atop a Raspberry Pi base or as a complete control solution on its own.

At its heart is an Atmel ATmega32U4 AVR microcontroller, which has 32 KB of flash program memory, 2.5 KB of RAM, and built-in USB functionality. The controller adds a complement of peripheral hardware useful for robotics applications, including dual H-bridge motor drivers capable of delivering at least 1.7 A per channel continuously, as well as an efficient switching voltage regulator and level shifters that enable it to power and communicate with a Raspberry Pi.

This A-Star is available in two versions with different operating voltage ranges: the Robot Controller LV (blue solder mask) accepts a 2.7 V to 11 V input voltage, and the Robot Controller SV (green solder mask) accepts a 5.5 V to 36 V input voltage.

Like other **A-Star programmable controllers** [<https://www.pololu.com/category/149/a-star-programmable-controllers>] (abbreviated A*), the robot controller features a USB interface and ships with a preloaded Arduino-compatible bootloader. We provide a software add-on that enables it to be easily programmed from the Arduino environment, as well as an Arduino library to help interface with its on-board hardware.



A-Star 32U4 Robot Controller LV with Raspberry Pi Bridge (SMT components only).



A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge (SMT components only).








A **USB A to Micro-B cable** [<https://www.pololu.com/product/2072>] (not included) is required to connect the A-Star 32U4 Robot Controller to a computer.

Features

- Dimensions: 65 mm × 56 mm (2.6" × 2.2")
- Programmable ATmega32U4 MCU with 32 KB flash, 2.5 KB SRAM, 1 KB EEPROM, and native full-speed USB (clocked by precision 16 MHz crystal oscillator)
- Preloaded with Arduino-compatible bootloader (no external programmer required)
- All 26 general-purpose I/O lines from the ATmega32U4 are broken out (including PB0, PD5, and PE2); 7 of these can be used as hardware PWM outputs and 12 of these can be used as analog inputs (some I/O lines are used by on-board hardware)
- Convenient 0.1"-spaced power, ground, and signal connection points
- Dual bidirectional motor drivers (≥1.7 A per channel)
- Buzzer option for simple sounds and music
- 3 user-controllable LEDs
- 3 user pushbuttons
- Reset button
- Level shifters for interfacing 5 V logic to 3.3 V Raspberry Pi
- Power features:
 - 5 V power can be sourced from USB or from external supply through on-board regulator (with several access points for connecting external power)
 - **LV**: 2.7 V to 11 V input
 - **SV**: 5.5 V to 36 V input
 - Switching 5 V regulator enables efficient operation
 - Power switch for external power inputs
 - Reverse-voltage protection on external power inputs
 - Power selection circuit allows for seamless switching between power sources while providing overcurrent protection, and feedback about which power source is selected
 - Provides 5 V power to Raspberry Pi
- 6-pin ISP header for use with an **external programmer** [<https://www.pololu.com/product/3172>]

A-Star comparison table

					
	<u>A-Star 328PB Micro</u>	<u>A-Star 32U4 Micro</u>	<u>A-Star 32U4 Mini ULV</u> <u>A-Star 32U4 Mini LV</u> <u>A-Star 32U4 Mini SV</u>	<u>A-Star 32U4 Prime LV</u> <u>A-Star 32U4 Prime SV</u>	<u>A-Star 32U4 Robot Controller LV</u> <u>A-Star 32U4 Robot Controller SV</u>
Microcontroller:	ATmega328PB		ATmega32U4		
User I/O lines:	24	18	26	26 ⁽¹⁾	26 ⁽¹⁾
PWM outputs:	9	7	7	7	7 ⁽¹⁾
Analog inputs:	8	8	12	12	12 ⁽¹⁾
Ground access points:	6	2	4	43	44
User LEDs:	1	2	3	3	3
User pushbuttons:	—	—	—	3	3
USB interface:		✓	✓	✓	✓
Reset button:	✓		✓	✓	✓
Power switch:				✓	✓
Buzzer option:				✓	✓
microSD option:				✓	
LCD option:				✓	
Motor drivers:					✓
Operating voltage:	3.3V VCC: 3.8 V to 15 V 5V VCC: 5.5 V to 15 V	5.5 V to 15 V	ULV: 0.5 V to 5.5 V LV: 2.7 V to 11.8 V SV: 5 V to 40 V	LV: 2 V to 16 V SV: 5 V to 36 V	LV: 2.7 V to 11 V SV: 5.5 V to 36 V
Regulator type:	3.3 V or 5 V linear	5 V linear	5 V switching ULV: step-up LV: step-up/step-down SV: step-down	5 V switching LV: step-up/step-down SV: step-down	5 V switching LV: step-up/step-down SV: step-down
Regulated current:⁽²⁾	100 mA	100 mA	ULV: 500 mA LV: 1 A SV: 800 mA	LV: 1.8 A SV: 1 A	LV: 1 A SV: 1.5 A
Dimensions:	1.3" × 0.7"	1" × 0.6"	1.9" × 0.7"	2.8" × 2.1"	2.6" × 2.2"
Weight:	1.5 g ⁽³⁾	1.3 g ⁽³⁾	3.4 g ⁽³⁾	13 g to 33 g	14 g to 23 g

¹ Some microcontroller resources are used by on-board hardware.

² These values are rough approximations for comparison purposes. Available current depends on input voltage, current consumed by the board, ambient conditions, and regulator topology. See product documentation and performance graphs for details.

³ Without included optional headers.

1.1. Supported operating systems

The A-Star 32U4 Robot Controller can be programmed using any operating system that supports the Arduino environment. We have tested the A-Stars, our Arduino software add-on, and the Arduino IDE on Microsoft Windows 10, 8.1, 8, 7, Vista, XP (with Service Pack 3), Linux, and Mac OS X.

2. Contacting Pololu

We would be delighted to hear from you about any of your projects and about your experience with the Pololu A-Star Robot Controller. You can **contact us** [<https://www.pololu.com/contact>] directly or post on our **forum** [<http://forum.pololu.com/>]. Tell us what we did well, what we could improve, what you would like to see in the future, or anything else you would like to say!

3. A-Star 32U4 Robot Controller with Raspberry Pi Bridge

3.1. Microcontroller

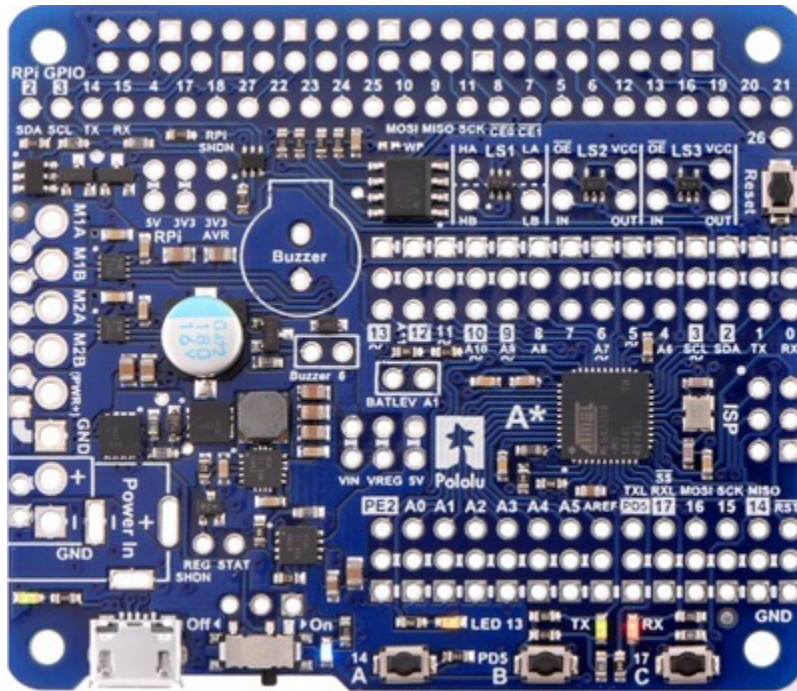
Like other **A-Star 32U4 programmable controllers** [<https://www.pololu.com/category/149/a-star-programmable-controllers>], the A-Star 32U4 Robot Controller features an integrated, USB-enabled ATmega32U4 AVR microcontroller from Atmel, clocked by a precision 16 MHz crystal oscillator. This is the same microcontroller and clock frequency used in the **Arduino Leonardo** [<https://www.pololu.com/product/2192>] and **Micro** [<https://www.pololu.com/product/2188>].

The board includes a USB Micro-B connector that can be used to connect to a computer's USB port via a **USB A to Micro-B cable** [<https://www.pololu.com/product/2072>] (not included). The USB connection can be used to transmit and receive data from the computer and program the board over USB. The USB connection can also provide power for the microcontroller and most of the other hardware on the A-Star (but not motor power); see **Section 3.5** for more details.

The robot controller's ATmega32U4 comes preloaded with the Arduino-compatible **A-Star 32U4 USB bootloader** [<https://www.pololu.com/docs/0J66/7>], which allows it to be easily programmed using the Arduino IDE. For more information about programming the A-Star 32U4 Robot Controller, see **Section 4**.

The board also has a 6-pin ISP header that allows it to be programmed with an external programmer, such as our **USB AVR programmer** [<https://www.pololu.com/product/3172>]. Pin 1 of the header is indicated with a small white dot and has an octagonal shape.

3.2. User interface



LEDs

The A-Star 32U4 Robot Controller has five indicator LEDs.

- A **yellow** user LED is connected to Arduino digital pin 13, or PC7. You can drive this pin **high** in a user program to turn this LED on. The **A-Star 32U4 Bootloader** [<https://www.pololu.com/docs/0J61/9>] fades this LED on and off while it is waiting for a sketch to be loaded.
- A **green** user LED is connected to PD5 and lights when the pin is driven **low**. While the board is running the A-Star 32U4 Bootloader or a program compiled in the Arduino environment, it will flash this LED when it is *transmitting* data via the USB connection.
- A **red** user LED is connected to Arduino pin 17, or PB0, and lights when the pin is driven **low**. While the board is running the A-Star 32U4 Bootloader or a program compiled in the Arduino environment, it will flash this LED when it is *receiving* data via the USB connection.

The AStar32U4 library contains functions that make it easier to control the three user LEDs. The green and red user LEDs also share I/O lines with pushbuttons (see below).

- A **blue** power LED next to the power switch indicates when the controller is receiving power from an external power source connected to the Power In terminals (the power switch must be turned on).

- A **green** power LED on the left edge of the board near the USB connector indicates when the USB bus voltage (VBUS) is present.

Pushbuttons

The A-Star 32U4 Robot Controller has four pushbuttons: a **reset button** on the right edge and **three user pushbuttons** located along the bottom edge of the main board. The user pushbuttons, labeled A, B, and C, are on Arduino pin 14 (PB3), PD5, and Arduino pin 17 (PB0), respectively. Pressing one of these buttons pulls the associated I/O pin to ground through a resistor.

The three buttons' I/O lines are also used for other purposes: pin 14 is MISO on the SPI interface, and PD5 and pin 17 control the green and red user LEDs. Although these uses require the pins to be driven by the AVR (or SPI slave devices in the case of MISO), resistors in the button circuits ensure that the A* will not be damaged even if the corresponding buttons are pressed at the same time, nor will SPI communications be disrupted. The functions in the AStar32U4 library take care of configuring the pins, reading and debouncing the buttons, and restoring the pins to their original states.

Buzzer

The assembled version of the A-Star 32U4 Robot Controller features a **buzzer** [<https://www.pololu.com/product/1484>] that can be used to generate simple sounds and music. The buzzer is not present on the SMT-only version, but the buzzer driver circuit is still populated, allowing you to solder in your own buzzer or speaker.

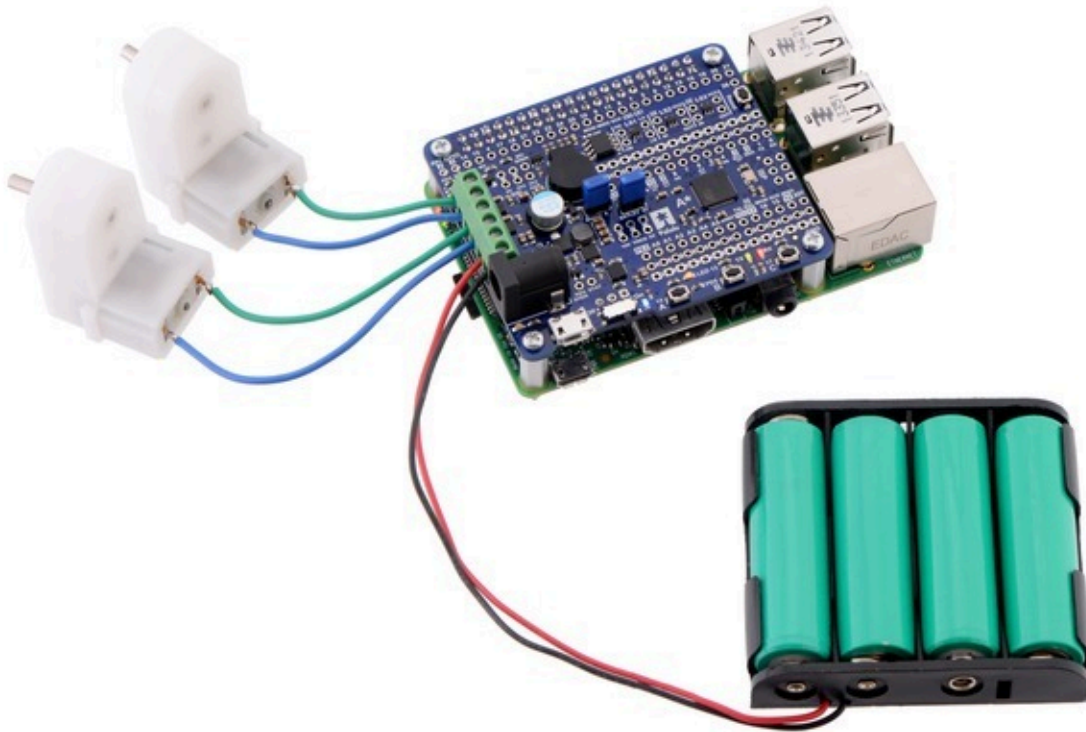
A through-hole jumper next to the buzzer provides a way to connect the buzzer input, labeled **Buzzer**, to digital pin 6 (which also serves as OC4D, a hardware PWM output from the AVR's 10-bit Timer4). If you alternate between driving the buzzer pin high and low at a given frequency, the buzzer will produce sound at that frequency. You can play notes and music with the buzzer using functions in the AStar32U4 library.

3.3. LV motor drivers

The **A-Star 32U4 Robot Controller LV** features two Texas Instruments DRV8838 motor drivers that can each deliver a continuous 1.8 A. The motor power supply is derived from an external source connected to the A-Star's Power In terminals, which also feeds the on-board voltage regulator and should have a voltage between 2.7 V and 11 V. (Although the DRV8838 works with motor supply voltages as low as 0 V, the voltage regulator requires a minimum input voltage of 2.7 V.) For more information about the drivers, see the **DRV8838 datasheet** [<https://www.pololu.com/file/0J806/drv8838.pdf>] (1MB pdf). (We also sell a standalone **carrier board** [<https://www.pololu.com/product/2990>] for this motor driver.)

The motor driver outputs, together with power pins, are connected to sets of through-holes along the left side of the board. The larger holes are designed for **3.5 mm screw terminal blocks**

[<https://www.pololu.com/product/2444>], which are soldered in on the assembled version. Alternatively, **0.1" headers** [<https://www.pololu.com/product/965>] can be used with the smaller holes on the SMT-only version, or motor wires can be soldered directly to either set of holes.



Driving motors with an A-Star 32U4 Robot Controller LV with Raspberry Pi Bridge on a Raspberry Pi Model B+ or Pi 2 Model B.

Although the LV and SV versions of the robot controller use different driver ICs, the motor driver interface is identical between the two versions. Four pins are used to control the drivers:

- **Digital pin 12**, or PD6, controls the **direction of motor 1** (when LOW, motor current flows from A to B; when HIGH, current flows from B to A).
- **PE2** (which does not have an Arduino pin number) controls the **direction of motor 2**.
- **Digital pin 9**, or PB5, controls the **speed of motor 1** with PWM (pulse width modulation) generated by the ATmega32U4's Timer1.
- **Digital pin 10**, or PB6, controls the **speed of motor 2** with PWM.

The AStar32U4 library provides functions that allow you to easily control the motors.

Maximum current and power dissipation considerations

The DRV8838 datasheet recommends a maximum continuous current of 1.8 A, and the robot controller's printed circuit board typically draws enough heat out of the motor driver chips to allow it to sustain this amount of current. In our tests, we found that the chip was able to deliver 1.8 A for many minutes without triggering a thermal shutdown. Our tests were conducted at 100% duty cycle with no forced air flow; PWMing the motor will introduce additional heating proportional to the frequency.

Motor currents over 1.8 A will likely trip either the over-temperature or over-current protection of the DRV8838. The current limiting threshold can be as low as 1.9 A or as high as 3.5 A. This limiting helps prevent damage to the drivers if a motor draws too much current, although the reduced current output will degrade motor performance.

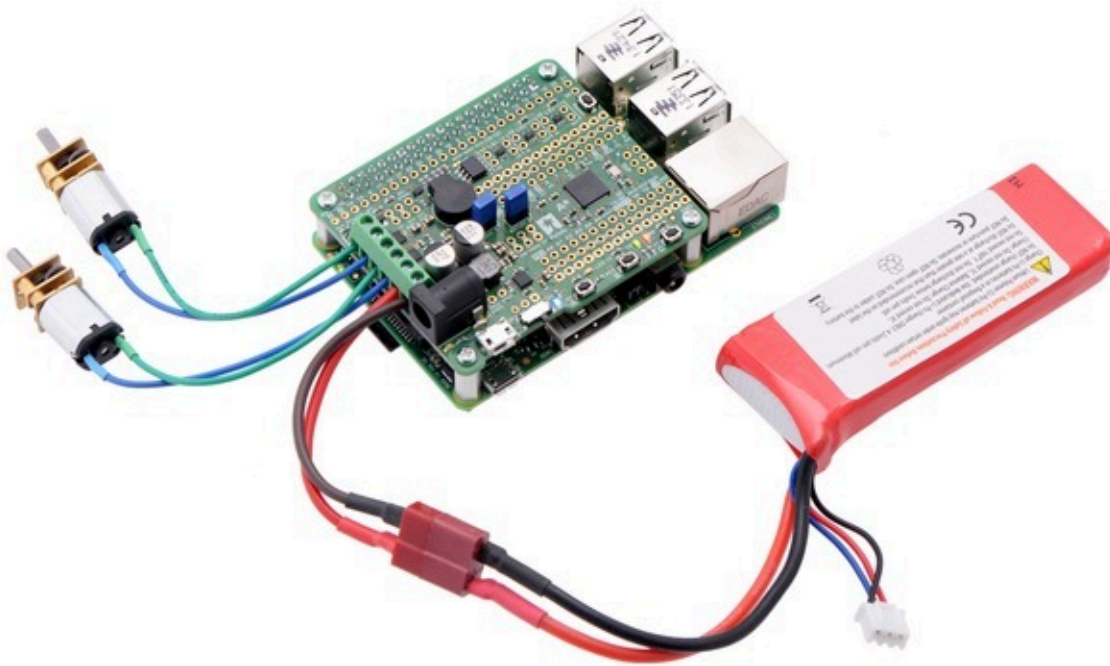
Some situations can increase heating of the motor drivers and reduce their maximum available current output, such as drawing a large amount of current from the A-Star's 5 V regulator, mounting the board on a Raspberry Pi, or using it in an enclosure that limits air flow. The actual current you can deliver will depend on how well you can keep the motor drivers cool.

This product can get **hot** enough to burn you long before the chip overheats. Take care when handling this product and other components connected to it.

3.4. 5V motor drivers

The **A-Star 32U4 Robot Controller 5V** features two Maxim MAX14870 motor drivers that can each deliver a continuous 1.7 A (2.5 A peak). The motor power supply is derived from an external source connected to the A-Star's Power In terminals, which also feeds the on-board voltage regulator and should have a voltage between 5.5 V and 36 V. (Although the MAX14870 works with motor supply voltages as low as 4.5 V, the voltage regulator requires a minimum input voltage of 5.5 V.) For more information about the drivers, see the **MAX14870 datasheet** [<https://www.pololu.com/file/0J885/MAX14870.pdf>] (492k pdf). (We also sell a standalone **carrier board** [<https://www.pololu.com/product/2961>] for this motor driver.)

The motor driver outputs, together with power pins, are connected to sets of through-holes along the left side of the board. The larger holes are designed for **3.5 mm screw terminal blocks** [<https://www.pololu.com/product/2444>], which are soldered in on the assembled version. Alternatively, **0.1" headers** [<https://www.pololu.com/product/965>] can be used with the smaller holes on the SMT-only version, or motor wires can be soldered directly to either set of holes.



Driving motors with an A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge on a Raspberry Pi Model B+ or Pi 2 Model B.

Although the LV and SV versions of the robot controller use different driver ICs, the motor driver interface is identical between the two versions. Four pins are used to control the drivers:

- **Digital pin 12**, or PD6, controls the **direction of motor 1** (when LOW, motor current flows from A to B; when HIGH, current flows from B to A).
- **PE2** (which does not have an Arduino pin number) controls the **direction of motor 2**.
- **Digital pin 9**, or PB5, controls the **speed of motor 1** with PWM (pulse width modulation) generated by the ATmega32U4's Timer1.
- **Digital pin 10**, or PB6, controls the **speed of motor 2** with PWM.

The AStar32U4 library provides functions that allow you to easily control the motors.

Maximum current and power dissipation considerations

The MAX14870 datasheet recommends a maximum current of 2.5 A. However, the chip by itself will typically overheat at lower currents. In our tests, we found that the chip was able to deliver 2.5 A for only a few seconds before the chip's thermal protection kicked in and disabled the motor outputs; a continuous current of 1.7 A was sustainable for many minutes without triggering a thermal shutdown. Our tests were conducted at 100% duty cycle with no forced air flow; PWMing the motor will introduce

additional heating proportional to the frequency.

Some situations can increase heating of the motor drivers and reduce their maximum available current output, such as drawing a large amount of current from the A-Star's 5 V regulator, mounting the board on a Raspberry Pi, or using it in an enclosure that limits air flow. The actual current you can deliver will depend on how well you can keep the motor driver cool. The MAX14870 also has over-current protection that triggers when the output current exceeds a certain limit (3 A minimum), which can help prevent damage to the drivers if a motor draws too much current.

This product can get **hot** enough to burn you long before the chip overheats. Take care when handling this product and other components connected to it.

3.5. Power

The A-Star 32U4 Robot Controller can be powered from an external voltage source, which is regulated to 5 V by its on-board switching regulator; the provided voltage is also used to supply the motor drivers directly. Alternatively, the A-Star's USB connector can be used to provide 5 V only, leaving the motor drivers unpowered.

An external supply should be connected to the points labeled **Power In** (or PWR+ and GND). These are power inputs with reverse-voltage protection that are controlled by the power switch. Together with the motor outputs, these power inputs are connected to sets of through-holes along the left side of the board. The larger holes are designed for **3.5 mm screw terminal blocks** [<https://www.pololu.com/product/2444>], while the smaller ones can be used with **0.1" headers** [<https://www.pololu.com/product/965>]. A **DC barrel jack** [<https://www.pololu.com/product/1139>] can also be used to provide power. On the assembled version of the board, a six-pin strip of 3.5 mm terminal blocks and a barrel jack are soldered in for motor and power connections.

Warning: You must never connect different power sources to multiple **Power In** locations at the same time, as doing so will create a short between the supplies.

The slide switch on the A* controls whether the external source is connected to the 5 V regulator and motor drivers, providing a convenient way to switch off external power to the robot controller (and optionally an attached Raspberry Pi) without unplugging any connections. The adjacent set of three pins provides a place to connect your own power switch: to enable external power, connect the middle pin to ground (accessible through the pin on the right).

When power is supplied through the Power In pins, the **VIN** pins can be used as an output to supply reverse-protected and switched power to other devices. Alternatively, the external supply can

be connected directly between VIN and GND, bypassing the reverse-voltage protection and power switch.

One of the positive power inputs, labeled (**PWR+**), can optionally be reconfigured to serve as a VIN access point instead. To do so, cut the surface-mount jumper on the underside of the board to disconnect the PWR+ pad and the center pad, then use solder to bridge the VIN pad and the center pad.

In a battery-powered application, it might be useful for the A-Star to monitor the battery's voltage level. The **BATLEV** pin provides access to a voltage divider that outputs one-third of the VIN voltage, and this voltage can be read by connecting it to the adjacent analog pin 1 (A1) (or another analog input). The `readBatteryMillivoltsLV()` and `readBatteryMillivoltsSV()` functions in the `AStar32U4` library can be used to determine the battery voltage from this reading.

5V regulator

VIN supplies power to a 5 V regulator, whose output is designated **VREG**. The allowable input voltage range depends on the particular version of the A-Star 32U4 Robot Controller:

- **LV**: 2.7 V to 11 V (see **Section 3.6** for regulator details)
- **SV**: 5.5 V to 36 V (see **Section 3.7** for regulator details)

The regulator can be disabled by driving the regulator shutdown pin, **REGSHDN**, high; this will cause 5 V power to be sourced from USB instead if it is available.

Power selection

The A-Star 32U4 Robot Controller's power selection circuit uses the **TPS2113A power multiplexer** [<https://www.pololu.com/product/2596>] from Texas Instruments to choose whether its 5 V supply (designated **5V**) is sourced from USB or an external supply via the regulator, allowing both sources to be connected at the same time and enabling the A* to safely and seamlessly transition between them. The TPS2113A is configured to select external power unless the regulator output falls below about 4.5 V. If this happens, it will select the higher of the two sources, which will typically be the USB 5 V bus voltage if the A* is connected to USB.

Consequently, when the A-Star is connected to a computer via USB, it will receive 5 V logic power even when the power switch is off. This can be useful if you want to upload or test a program without drawing power from batteries and without operating motors. It is safe to have USB connected and external power switched on at the same time.

The currently selected source is indicated by the **STAT** pin in the middle of the board; this pin is an open-drain output that is low if the external power source is selected and high-impedance if the

USB supply is selected. The current limit of the TPS2113A is set to about 1.9 A nominally. For more information about the power multiplexer, see the **TPS2113A datasheet** [<https://www.pololu.com/file/0J771/tps2113a.pdf>] (1MB pdf).

Raspberry Pi power

By default, the robot controller will provide power from its 5V line to an attached Raspberry Pi. In this situation, we recommend supplying external power through the Power In pins so that the Raspberry Pi receives power from the A-Star's on-board switching regulator. Alternatively, you can use a **USB wall power adapter** [<https://www.pololu.com/product/1459>] to supply power through the A-Star's USB connector, although we have sometimes observed AVR brown-out resets occurring when the controller is powering the Raspberry Pi this way. A typical computer USB port might not be able to supply enough current to properly power the A-Star and an attached Raspberry Pi.

Power provided to the Raspberry Pi can be switched off by driving the Raspberry Pi shutdown pin, **RPISHDN**, to 5 V.

An ideal diode circuit on the A* makes it safe to have a different power supply connected to the Raspberry Pi (for example, through the Raspberry Pi's USB Micro-B receptacle) while the A-Star 32U4 Robot Controller is connected and powered. (In other words, it is safe to have any combination of A-Star USB power, A-Star external power, and Raspberry Pi USB power connected to the system.) The **RPI5V** pin provides direct access to the Raspberry Pi's 5 V line, which will typically come from the higher of the two power sources.

Note that the diode circuit prevents power from being shared in the reverse direction: the Raspberry Pi cannot supply 5 V logic power to the robot controller through the 40-pin connector.

3.3V supply for level shifters

Some of the level shifters on the A-Star are supplied with 3.3 V provided by an attached Raspberry Pi, and this voltage is also accessible through the **RPI3V3** pins. If you want to use the level shifters in an application without a Raspberry Pi, you can instead supply them from the ATmega32U4 microcontroller's integrated 3.3 V regulator by connecting the **AVR3V3** pin to the adjacent RPI3V3 pin. (It is preferable to use the Raspberry Pi's 3.3 V supply when available because it is produced by a switching regulator that is more efficient and can provide more current than the linear regulator in the AVR.) Alternatively, you can supply a different voltage between 1.65 V and 5 V to the RPI3V3 pin to make the level shifters shift to that voltage instead. See **Section 3.8** for information about the level shifters.

3.6. LV regulator

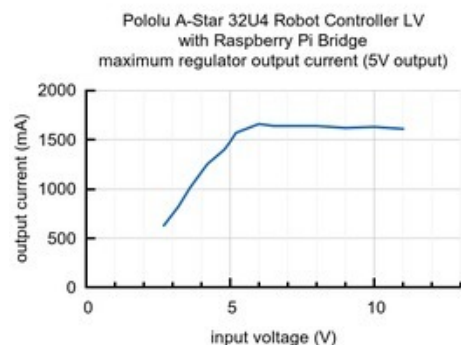
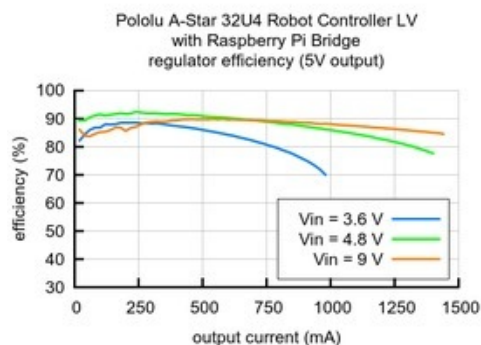


The **A-Star 32U4 Robot Controller LV** can be powered from a **2.7 V to 11 V** external source. The input voltage is regulated to 5 V by a TPS63061 switching step-up/step-down (buck-boost) converter from Texas Instruments; although the regulator itself works with input voltages up to 11.8 V, the motor drivers limit the robot controller's maximum input voltage to 11 V. (We also make a **standalone regulator** [<https://www.pololu.com/product/2123>] based on this integrated circuit.)

The regulator's flexibility in input voltage is especially well-suited for battery-powered applications in which the battery voltage begins above 5 V and drops below 5 V as the battery discharges. Without the typical restriction on the battery voltage staying above 5 V throughout its life, a wider range of battery types can be considered. For example:

- A 4-cell battery holder, which might have a 6 V output with fresh alkalines or a 4.0 V output with partially discharged NiMH cells, can be used to power this A*.
- A disposable 9 V battery powering the board can be discharged to under 3 V instead of cutting out at 6 V, as with typical linear or step-down regulators.

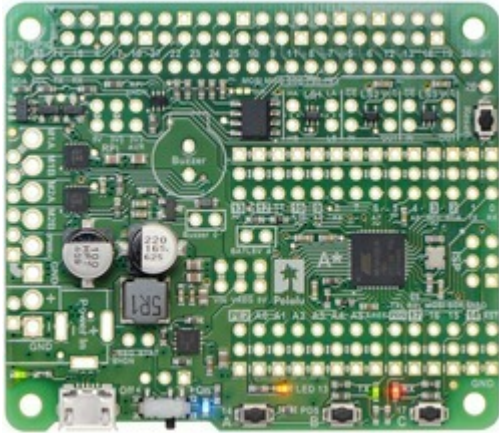
As shown in the left graph below, the LV's switching regulator has an efficiency – defined as (Power out)/(Power in) – of 80% to 90% for most combinations of input voltage and load.



The A-Star's components, including the microcontroller and LEDs, draw 30 mA to 40 mA in typical applications (without the buzzer). The rest of the regulator's achievable output current, which depends on input voltage as well as ambient conditions, can be used to power other devices; this can include an attached Raspberry Pi (which typically draws a few hundred milliamps). The right graph above shows output currents at which the voltage regulator's over-temperature protection typically kicks in after a few seconds. These currents represent the limit of the regulator's capability and cannot be sustained

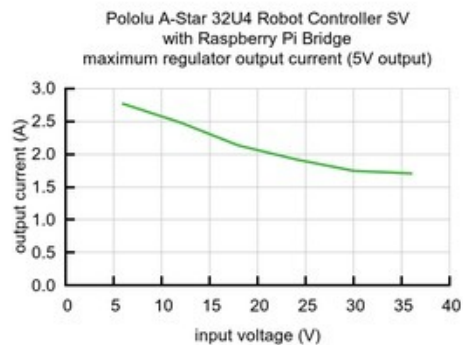
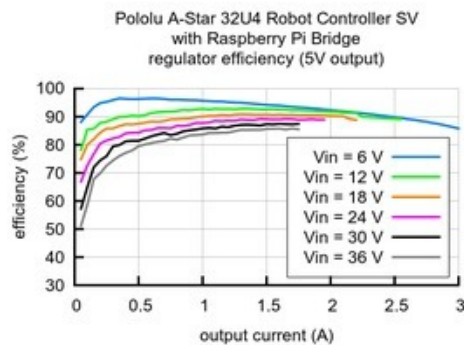
for long periods; under typical operating conditions, a safe limit for the maximum continuous regulator output current is 60% to 70% of the values shown in the graph.

3.7. 5V regulator



The **A-Star 32U4 Robot Controller SV** can be powered from a **5.5 V to 36 V** external source. The input voltage is regulated to 5 V by an MP4423H switching step-down (buck) converter from Monolithic Power Systems. (We also make a **standalone regulator** [<https://www.pololu.com/product/2858>] based on this integrated circuit.)

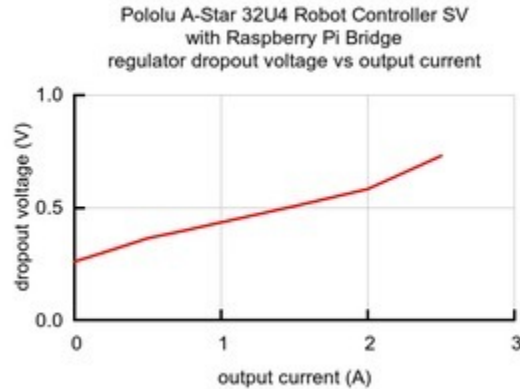
As shown in the left graph below, the SV's switching regulator has an efficiency – defined as (Power out)/(Power in) – of 80% to 95% for most combinations of input voltage and load.



The A-Star's components, including the microcontroller and LEDs, draw 30 mA to 40 mA in typical applications (without the buzzer). The rest of the regulator's achievable output current, which depends on input voltage as well as ambient conditions, can be used to power other devices; this can include an attached Raspberry Pi (which typically draws a few hundred milliamps). The right graph above shows the output currents where the regulator's output voltage drops below 4.75 V. These currents are close to the limits of the regulator's capability and generally cannot be sustained for long periods; under typical operating conditions, a safe limit for the maximum continuous regulator output current is 60% to 70% of the values shown in the graph.

The dropout voltage of a step-down regulator is defined as the minimum amount by which the input voltage must exceed the regulator's target output voltage in order to assure the target output can be achieved. As can be seen in the graph below, the dropout voltage of the Robot Controller SV's regulator increases approximately linearly with the output current. For light loads where the dropout

voltage is small, the board can operate almost down to 5 V. However, for larger loads, the dropout voltage should be taken into consideration when selecting a power supply; operating above 6 V will ensure the full output current is available.

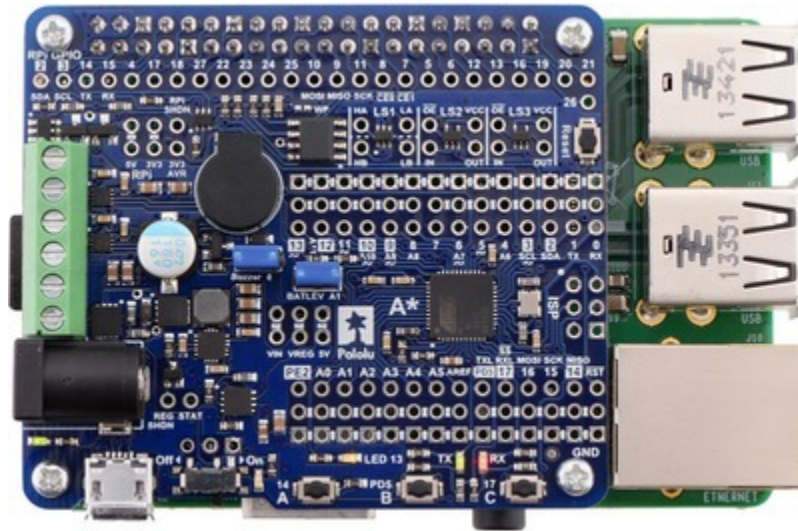


Typical dropout voltage of the 5 V regulator on the A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge.

Note: Although the MP4423H is rated for a maximum operating input voltage of 36 V, it is *not* appropriate to power the Robot Controller SV with a 36 V battery, as battery voltages can be much higher than nominal voltages when they are charged. The maximum nominal battery voltage we recommend is 24 V, and if you approach that limit, you should take extra precautions to prevent LC voltage spikes from damaging the board (see **this application note** [<https://www.pololu.com/docs/0J16>] for more information). A good practice is to ensure that the A-Star's power switch is off before connecting it to a voltage source.

3.8. Raspberry Pi interface and level shifters

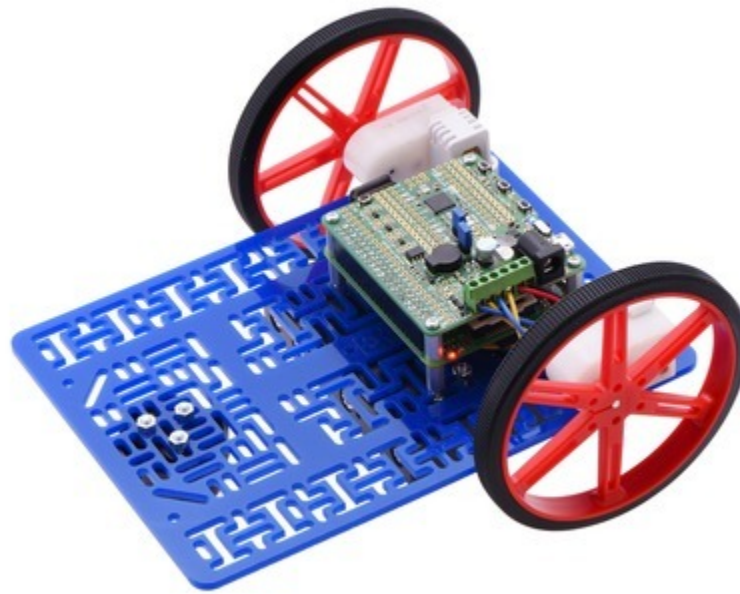
The A-Star 32U4 Robot Controller with Raspberry Pi Bridge can be used as an expansion board on top of a Raspberry Pi single-board computer. It conforms to the Raspberry Pi HAT (Hardware Attached on Top) specification and is designed to connect to the Model B+ and newer versions of the Raspberry Pi with 40-pin GPIO headers (including the **Raspberry Pi 3 Model B** [<https://www.pololu.com/product/2759>] and **Model A+** [<https://www.pololu.com/product/2760>]). A **2×20-pin 0.1" female header** [<https://www.pololu.com/product/1037>] is soldered to the assembled version of the robot controller, and it ships with a set of four **standoffs** [<https://www.pololu.com/product/1952>], **screws** [<https://www.pololu.com/product/1968>], and **nuts** [<https://www.pololu.com/product/1967>]. (The header and mounting hardware are not included with the SMT-only version, but you can solder in either a standard or **stackable** [<https://www.pololu.com/product/2748>] header yourself.)



I²C communication

When used as a Raspberry Pi add-on, the A-Star is designed to serve as an auxiliary controller, communicating with the Raspberry Pi using an I²C interface (also known as 2-wire Serial Interface, or TWI). As such, the ATmega32U4 microcontroller's I²C data and clock lines (SDA and SCL) are connected to the corresponding lines on the Raspberry Pi's I²C bus 1 through on-board level-shifting circuits. These bidirectional level shifters convert between the AVR's 5 V logic level and the Raspberry Pi's 3.3 V logic level.

We have written an **Arduino library** [<https://github.com/pololu/pololu-rpi-slave-arduino-library>] for the robot controller that lets it act as an I²C slave and provides a framework for communication between the A-Star and a Raspberry Pi master. A **tutorial** [<https://www.pololu.com/blog/577>] on the Pololu blog demonstrates this library and its included example code, using them to make a robot that can be remotely controlled and monitored through a web server running on the Raspberry Pi.

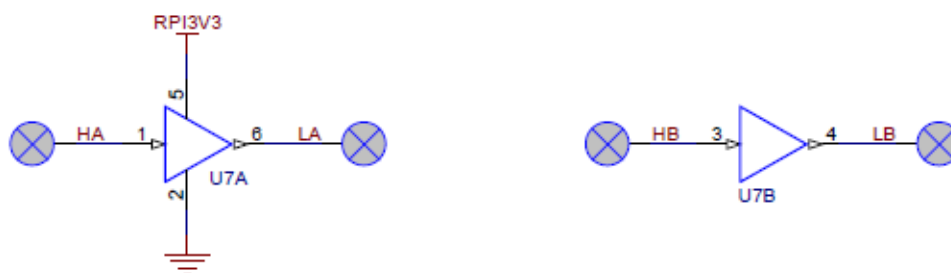


Raspberry Pi robot using the A-Star 32U4 Robot Controller.

General-purpose level shifters

In addition to the dedicated I²C level shifters, the A-Star board also makes available a few general-purpose level shifters that are not connected to any signals by default.

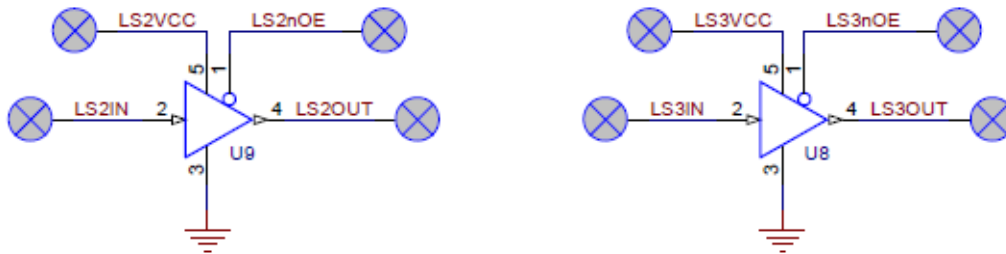
LS1 is a dual-channel unidirectional level shifter that converts a pair of 5 V inputs (**HA** and **HB**) to a pair of corresponding 3.3 V outputs (**LA** and **LB**).



LS2 and **LS3** are each single-channel, tristatable, unidirectional level shifters. Each of these exposes four pins: $\overline{\text{OE}}$ (output enable), **IN** (input), **OUT** (shifted output), and **VCC** (logic supply voltage).

- When $\overline{\text{OE}}$ is high, **OUT** is in a high impedance state.

- When \overline{OE} is low, OUT matches the state of IN, shifted to the voltage supplied on VCC.



For example, if you pull \overline{OE} low, connect a 3.3 V signal to IN, and connect 5V to VCC, the signal will be shifted to 5 V logic level on OUT.

The input logic level can be 1.8 V to 5.5 V, while VCC (and the output logic level) can be 3 V to 5.5 V. The IN signal can have either a lower or higher logic level than the VCC voltage: you could connect a 5 V signal to IN and a 3.3 V to VCC or a 3.3 V signal to IN and a 5 V to VCC.

Powering the Raspberry Pi from the robot controller

The robot controller will provide 5 V power to an attached Raspberry Pi by default, and 3.3 V from the Raspberry Pi will be used to supply some of the level shifters. See **Section 3.5** for more details about how power is shared and can be controlled between the two boards.

ID EEPROM

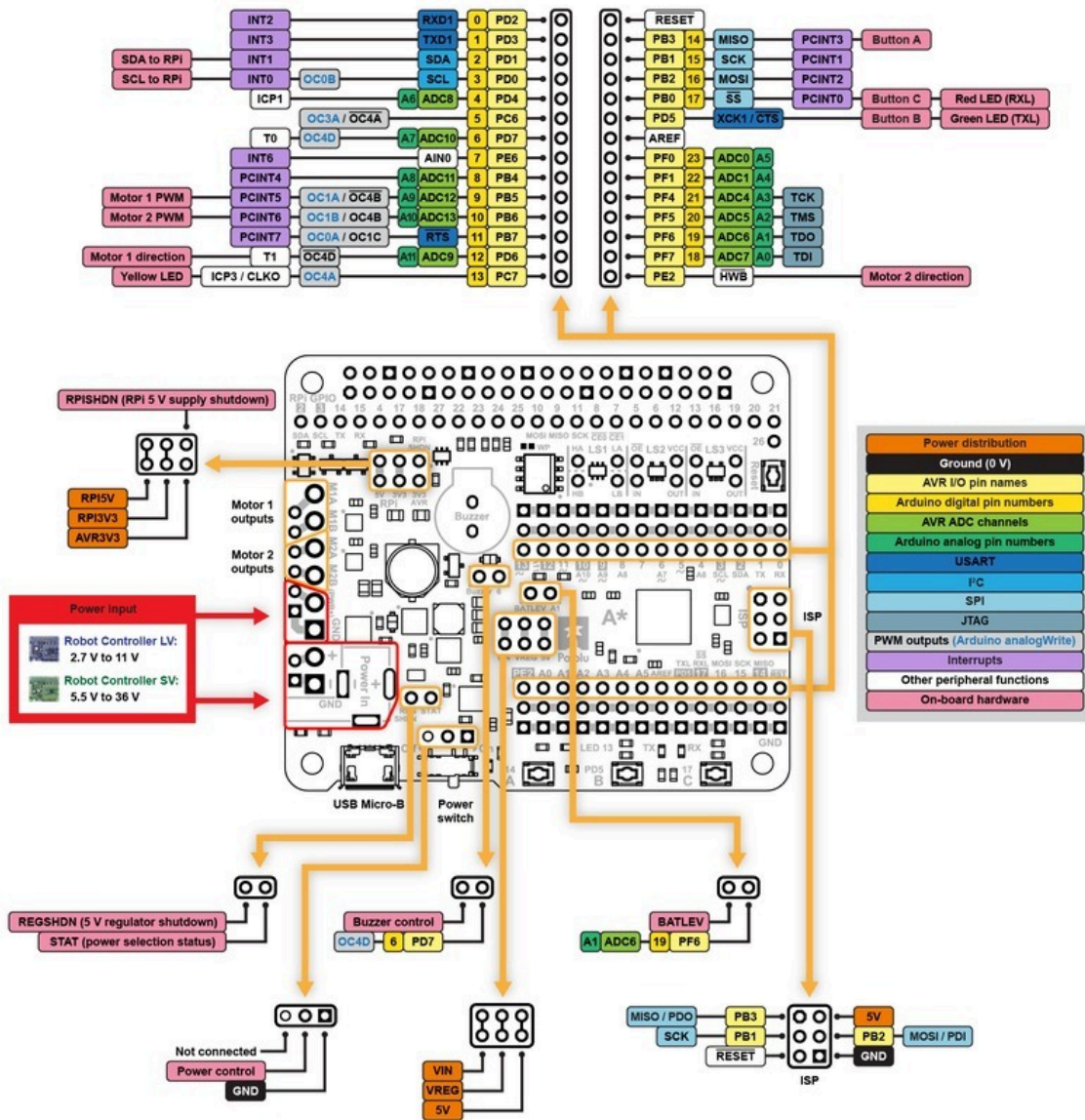
The A-Star board includes a 32-kilobit (4096-byte) EEPROM that connects to the Raspberry Pi's ID_SD and ID_SC pins. The EEPROM ships with its contents blank, but you can program it as an ID EEPROM in the format specified by the **Raspberry Pi HAT specifications** [<https://github.com/raspberrypi/hats>], using the utilities provided there. When suitably programmed, the EEPROM can help the Raspberry Pi identify and configure itself to work with the add-on board.

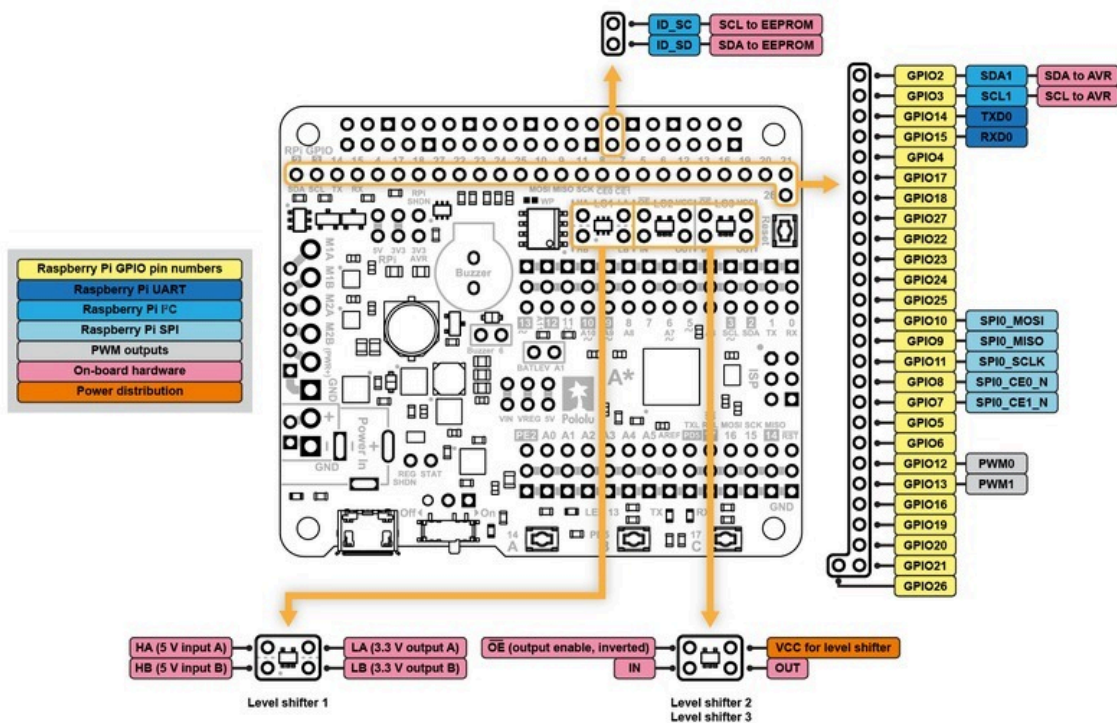
Write protection for the EEPROM can be enabled by using solder to bridge the surface-mount jumper labeled "WP" next to the EEPROM chip. (The EEPROM is not write-protected by default.)

3.9. Pinout

These diagrams identify the I/O and power pins on the A-Star 32U4 Robot Controller with Raspberry Pi Bridge (LV and SV versions); they are also available (along with the power distribution diagram below) as a **printable PDF** [<https://www.pololu.com/file/0J951/a-star-32u4-robot-controller-with-raspberry-pi-bridge-pinout-power.pdf>] (8MB pdf). For more information about the ATmega32U4 microcontroller and its

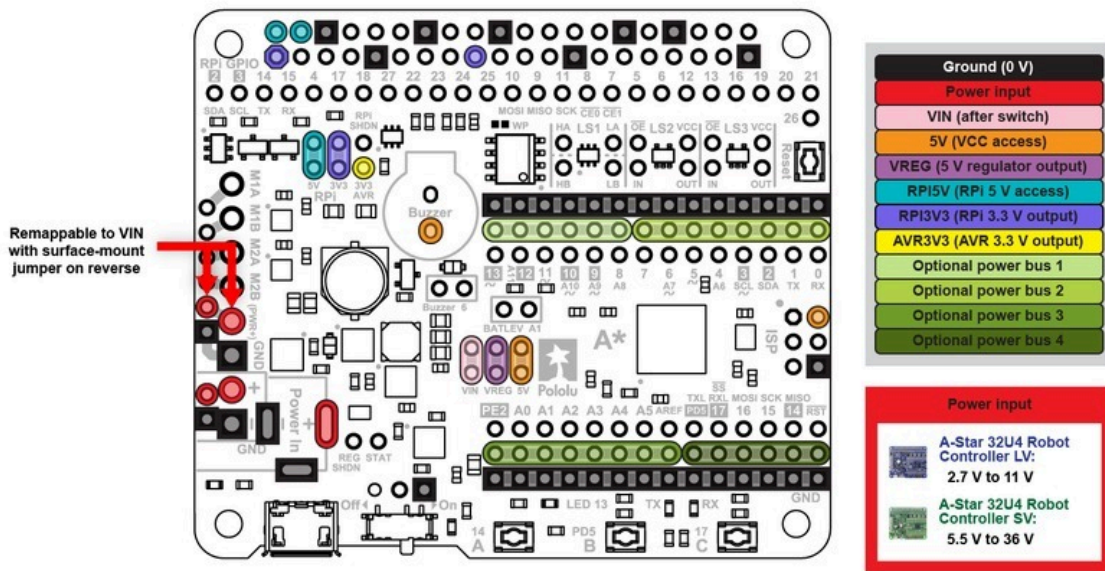
peripherals, see Atmel's ATmega32U4 documentation.





Printed on the A* circuit board are indicators that you can use to quickly identify each AVR I/O pin's capabilities: an analog pin number (prefixed by the letter A) means the pin can be used as an analog input, and a square wave symbol next to the hole pair means the pin can be used as a PWM output. Pin numbers printed as inverted text surrounded by a white box indicate that the pin is used by on-board hardware (see **Section 3.10** for more information about pin assignments).

Additional rows of power and ground pins run along the I/O pins to allow easy connections for devices like sensors and servos. The outermost row of pins is connected to ground, and the middle row is intended for power connections, but not connected by default. The power pins are split into four buses: two buses along the top, split between digital pins 7 and 8, and two buses along the bottom, split between AREF and PD5. You can configure each bus separately by using a jumper to connect it to a voltage source of your choice. (Access points for various supply voltages are available nearby.)



3.10. Pin assignments

The table below lists the most important pin assignments for the ATmega32U4 on the A-Star 32U4 Robot Controller with Raspberry Pi Bridge. This table is helpful if you want to add your own electronics to the robot controller, write your own low-level code for interfacing with the hardware, or just want to understand better how the A-Star works. Each row represents a physical pin on the ATmega32U4.

The “ATmega32U4 pin name” column shows the official name of the pin according to the **ATmega32U4 datasheet** [<https://www.microchip.com/wwwproducts/en/ATmega32u4>].

The “Arduino pin names” column lists the names provided by the Arduino environment for the pin. These names can generally be used as arguments to any function that takes a pin number. However, there are some exceptions. For example, passing the number 4 to `analogRead` actually reads pin A4, not pin 4. Also, due to hardware limitations, some functions only work on a limited set of pins.

The “A-Star 32U4 Robot Controller functions” column documents what the pin is used for on the A-Star 32U4 Robot Controller. Many pins can serve multiple purposes concurrently by switching modes. For example, PB0 can read the state of button C when it is an input, and it can control the red LED when it is an output.

The “Notes/alternate functions” column documents other features of the pin, although some of those features might be impractical to use.

ATmega32U4 pin name	Arduino pin names	A-Star 32U4 Robot Controller functions	Notes/alternate functions
PB3	14, MISO	User pushbutton A	SPI Master Input/Slave Output (MISO) Pin-change interrupt (PCINT3) ISP programming line (PDO)
PB0	17, SS	Red LED (RX) User pushbutton C	SPI slave select (\overline{SS}) Pin-change interrupt (PCINT0)
PC7	13	Yellow LED	Timer4 PWM output A (OC4A) Timer3 input capture pin (ICP3) Divided system clock output (CLKO)
PD5	-	Green LED (TX) User pushbutton B	UART external clock (XCK1) UART flow control (\overline{CTS})
PD6	12, A11, 29	Motor 1 direction	Analog input (ADC9) Timer4 PWM output D ($\overline{OC4D}$) Timer1 counter source (T1)
PB5	9, A9, 27	Motor 1 PWM	Analog input (ADC12) Timer1 PWM output A (OC1A) Timer4 PWM output B ($\overline{OC4B}$) Pin-change interrupt (PCINT5)
PE2	-	Motor 2 direction	Hardware bootloader select (\overline{HWB})
PB6	10, A10, 28	Motor 2 PWM	Analog input (ADC13) Timer1 PWM output B (OC1B) Timer4 PWM output B (OC4B) Pin-change interrupt (PCINT6)
PD0	3, SCL	I ² C clock for Raspberry Pi communication	Timer0 PWM output B (OC0B) External interrupt source ($\overline{INT0}$)
PD1	2, SDA	I ² C data for Raspberry Pi communication	External interrupt source ($\overline{INT1}$)
PD7	6, A7, 25	<i>Free I/O</i> <i>Jumper to buzzer PWM</i>	Analog input (ADC10) Timer4 PWM output D (OC4D) Timer0 counter source (T0)
PF6	A1, 19	<i>Free I/O</i> <i>Jumper to battery level input (VIN/3)</i>	Analog input (ADC6) JTAG test data out (TDO)
PD2	0	<i>Free I/O</i>	UART receive pin (RXD1) External interrupt source ($\overline{INT2}$)

ATmega32U4 pin name	Arduino pin names	A-Star 32U4 Robot Controller functions	Notes/alternate functions
PD3	1	<i>Free I/O</i>	UART transmit pin (TXD1) External interrupt source ($\overline{\text{INT3}}$)
PD4	4, A6, 24	<i>Free I/O</i>	Analog input (ADC8) Timer1 input capture pin (ICP1)
PC6	5	<i>Free I/O</i>	Timer3 PWM output A (OC3A) Timer4 PWM output A ($\overline{\text{OC4A}}$)
PE6	7	<i>Free I/O</i>	Analog comparator negative input (AIN0) External interrupt source (INT6)
PB4	8, A8, 26	<i>Free I/O</i>	Analog input (ADC11) Pin-change interrupt (PCINT4) Timer0 PWM output A (OC0A) Timer1 PWM output C (OC1C)
PB7	11	<i>Free I/O</i>	UART flow control ($\overline{\text{RTS}}$) Pin-change interrupt (PCINT7)
PB1	15, SCK	<i>Free I/O</i>	SPI Clock (SCK) Pin-change interrupt (PCINT1) ISP programming line (SCK)
PB2	16, MOSI	<i>Free I/O</i>	SPI Master Output/Slave Input (MOSI) Pin-change interrupt (PCINT2) ISP programming line (PDI)
PF7	A0, 18	<i>Free I/O</i>	Analog input (ADC7) JTAG test data in (TDI)
PF5	A2, 20	<i>Free I/O</i>	Analog input (ADC5) JTAG test mode select (TMS)
PF4	A3, 21	<i>Free I/O</i>	Analog input (ADC4) JTAG test clock (TCK)
PF1	A4, 22	<i>Free I/O</i>	Analog input (ADC1)
PF0	A5, 23	<i>Free I/O</i>	Analog input (ADC0)
$\overline{\text{RESET}}$	-	Reset pushbutton	internally pulled high, active low
AREF	-	-	Analog reference

3.11. AVR Timers

The ATmega32U4 has 4 timers: Timer0, Timer1, Timer3, and Timer4. Each timer has a different set of features, as documented in the datasheet.

- Timer0 is used by the Arduino environment for timing-related functions like `millis()`.
- Timer1 is used by the A-Star 32U4 Arduino library for driving motors.
- Timer3 is *not* used by the A-Star 32U4 Arduino library and can be freely used for your own purposes.
- Timer4 is used by the A-Star 32U4 Arduino library for controlling the buzzer. The **Buzzer** pin must be connected to the adjacent digital pin 6 (PD7; Timer4 output OC4D) for this to work.

3.12. Schematics and dimensions

Schematics

The schematic diagrams for the A-Star 32U4 Robot Controllers with Raspberry Pi Bridge are available as PDFs:

- **A-Star 32U4 Robot Controller LV with Raspberry Pi Bridge schematic diagram**
[<https://www.pololu.com/file/0J950/a-star-32u4-robot-controller-lv-with-raspberry-pi-bridge-schematic.pdf>]
(382k pdf)
- **A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge schematic diagram**
[<https://www.pololu.com/file/0J1112/a-star-32u4-robot-controller-sv-with-raspberry-pi-bridge-schematic.pdf>]
(427k pdf)

Dimensions

Dimension diagrams for the A-Star 32U4 Robot Controller are available as PDFs:

- **A-Star 32U4 Robot Controller LV with Raspberry Pi Bridge dimension diagram**
[<https://www.pololu.com/file/0J1123/a-star-32u4-robot-controller-lv-with-raspberry-pi-bridge-dimension-diagram.pdf>] (2MB pdf)
- **A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge dimension diagram**
[<https://www.pololu.com/file/0J1118/a-star-32u4-robot-controller-sv-with-raspberry-pi-bridge-dimension-diagram.pdf>] (2MB pdf)

DXF files for each version of the A-Star 32U4 Robot Controller, which show the main board outline along with the sizes and locations of all of the holes on the board are also available:

- **A-Star 32U4 Robot Controller LV with Raspberry Pi Bridge drill guide**

[<https://www.pololu.com/file/0J1121/ac04b-drill.dxf>] (188k dxf)

- **A-Star 32U4 Robot Controller SV with Raspberry Pi Bridge drill guide**

[<https://www.pololu.com/file/0J1122/ac04c-drill.dxf>] (197k dxf)

4. Getting started

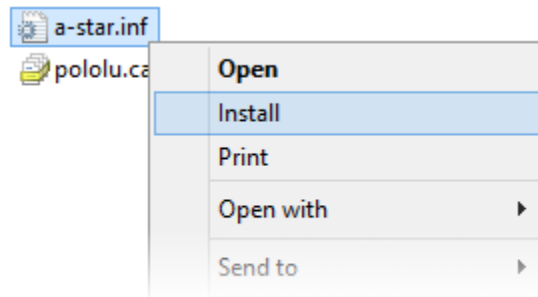
4.1. Installing Windows drivers



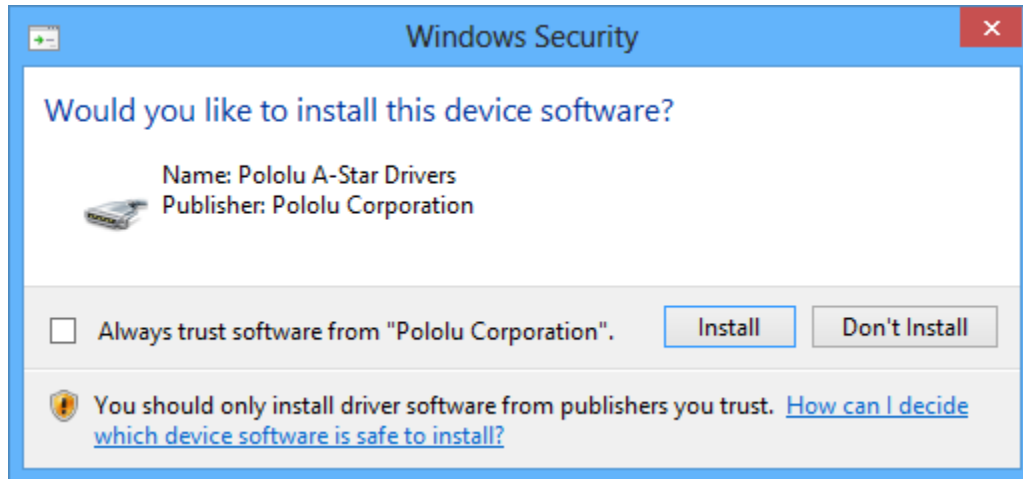
If you use Windows XP, you will need to have either Service Pack 3 or Hotfix KB918365 installed before installing the A-Star drivers. Some users who installed the hotfix have reported problems that were solved by upgrading to Service Pack 3, so we recommend Service Pack 3 over the hotfix.

Before you connect your Pololu A-Star 32U4 (or another of our 32U4 family of boards) to a computer running Microsoft Windows, you should install its drivers:

1. Download the **A-Star Windows Drivers** [<https://www.pololu.com/file/0J1240/a-star-windows-1.3.0.0.zip>] (7k zip) and extract the ZIP file to a temporary folder on your computer. (These files are also available in the “drivers” directory from the **A-Star repository on GitHub** [<https://github.com/pololu/a-star>].)
2. Open the “a-star-windows” folder. Right-click on “a-star.inf” and select “Install”.



3. Windows will ask you whether you want to install the drivers. Click “Install” (Windows 10, 8, 7, and Vista) or “Continue Anyway” (Windows XP).



4. Windows will not tell you when the installation is complete, but it should be done after a few seconds.

Windows 10, Windows 8, Windows 7, and Windows Vista users: After installing the drivers, your computer should automatically recognize the device when you connect it via USB. No further action from you is required. However, the first time you connect an A-Star device to your computer, Windows will take several seconds to recognize the device and configure itself properly. The first time you program the device, Windows will again take several seconds to recognize the A-Star USB bootloader, and this could cause the programming operation to fail the first time. Also, Windows will need to re-recognize the device and the bootloader if you connect the board to another USB port that it has not been connected to before.

Windows XP users: After installing the drivers, you will need to follow steps 5–9 for each new A-Star device you connect to your computer. You will also need to follow these steps the first time you attempt to program the device in order to make Windows recognize the bootloader, and when you connect the device to a different USB port that it has not been connected to before.

5. Connect the device to your computer's USB port.
6. When the "Found New Hardware Wizard" is displayed, select "No, not this time" and click "Next".
7. On the second screen of the "Found New Hardware Wizard", select "Install the software automatically" and click "Next".
8. Windows XP will warn you again that the driver has not been tested by Microsoft and recommend that you stop the installation. Click "Continue Anyway".
9. When you have finished the "Found New Hardware Wizard", click "Finish".

COM port details

After installing the drivers and plugging in an A-Star, in the “Ports (COM & LPT)” category of the Device Manager, you should see a COM port for the A-Star’s running sketch named “Pololu A-Star 32U4”.

You might see that the COM port is named “USB Serial Device” in the Device Manager instead of having a descriptive name. This can happen if you are using Windows 10 or later and you plugged the A-Star into your computer before installing our drivers for it. In that case, Windows will set up your A-Star using the default Windows serial driver (usbser.inf), and it will display “USB Serial Device” as the name for the port. The port will still be usable, but it will be hard to tell if it is the right one because of the generic name shown in the Device Manager. We recommend fixing the names in the Device Manager by right-clicking on each “USB Serial Device” entry, selecting “Update Driver Software...”, and then selecting “Search automatically for updated driver software”. Windows should find the drivers you already installed, which contain the correct name for the port.

If you are using Windows 10 or later and choose not to install the drivers, the A-Star will still be usable. To tell which “USB Serial Device” in your Device Manager is the A-Star, double-click on each one and look at the “Hardware Ids” property in the “Details” tab. An A-Star running a sketch will have the ID `USB\VID_1FFB&PID_2300&MI_00`, while an A-Star in bootloader mode will have the ID `USB\VID_1FFB&PID_0101`.

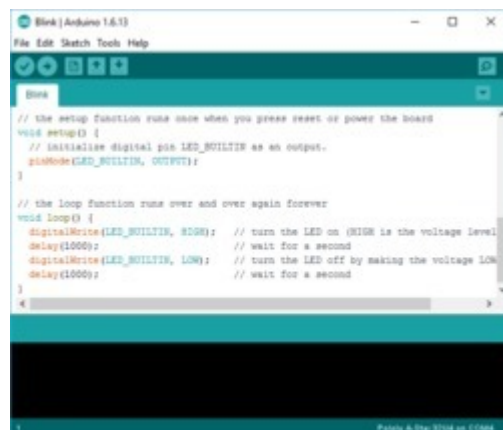
If you want to change the COM port numbers assigned to your A-Star, you can do so using the Device Manager. Double-click a COM port to open its properties dialog, and click the “Advanced...” button in the “Port Settings” tab.

4.2. Programming using the Arduino IDE

Our 32U4 family of boards can be programmed from the popular Arduino integrated development environment (IDE). The Arduino IDE is a cross-platform, open source application that integrates a C++ code editor, the GNU C++ compiler, and a program upload utility. To get started programming your device with the Arduino IDE (version 1.6.4 or later), follow these steps:

Download the Arduino IDE from the **Arduino Download page** [<http://arduino.cc/en/Main/Software>], install it, and start it.

1. In the Arduino IDE, open the **File** menu (Windows/Linux) or the **Arduino** menu (macOS) and select “Preferences”.

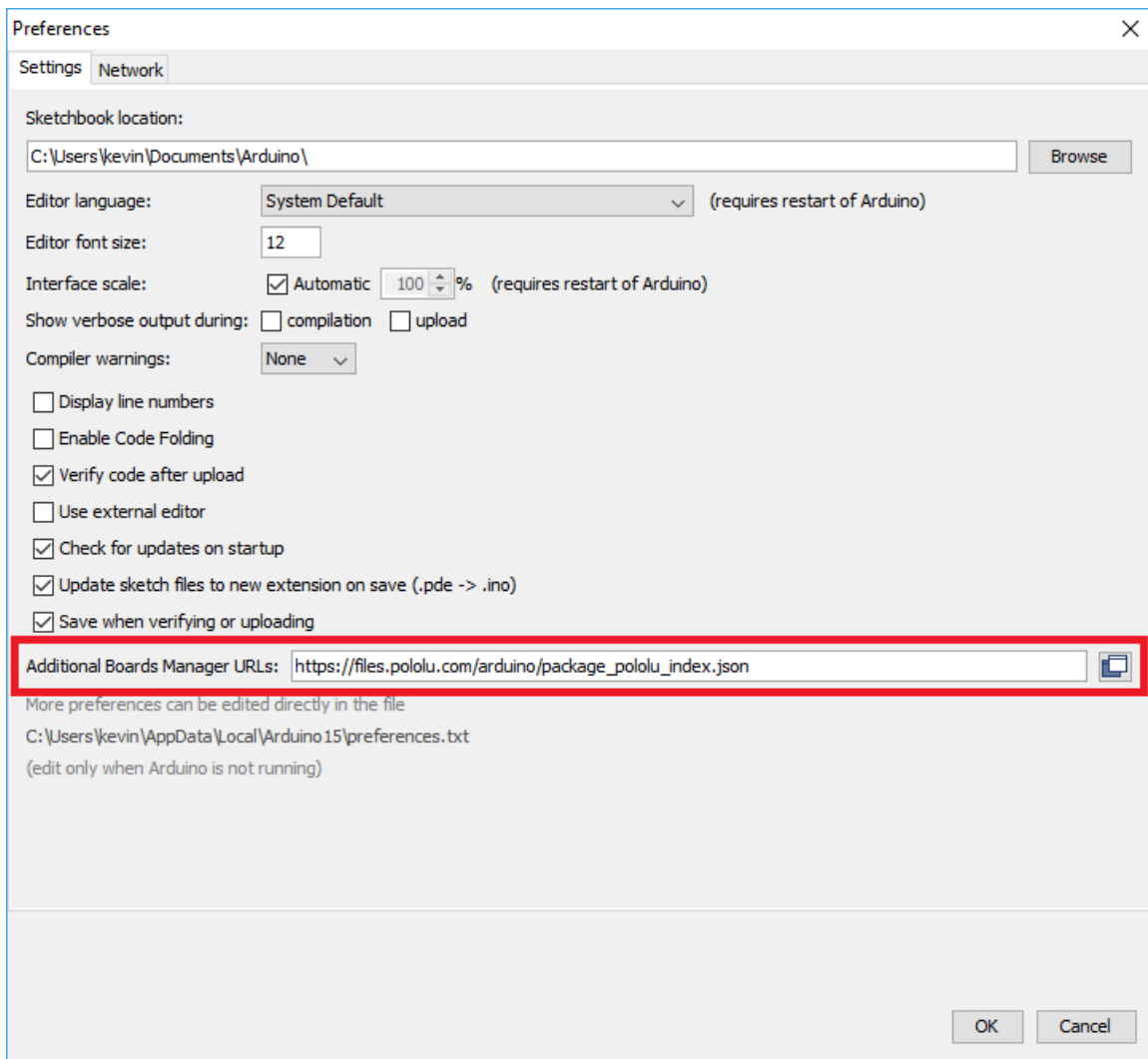


Programming the A-Star 32U4 from the Arduino IDE.

2. In the Preferences dialog, find the “Additional Boards Manager URLs” text box (highlighted in the picture below). Copy and paste the following URL into this box:

https://files.pololu.com/arduino/package_pololu_index.json

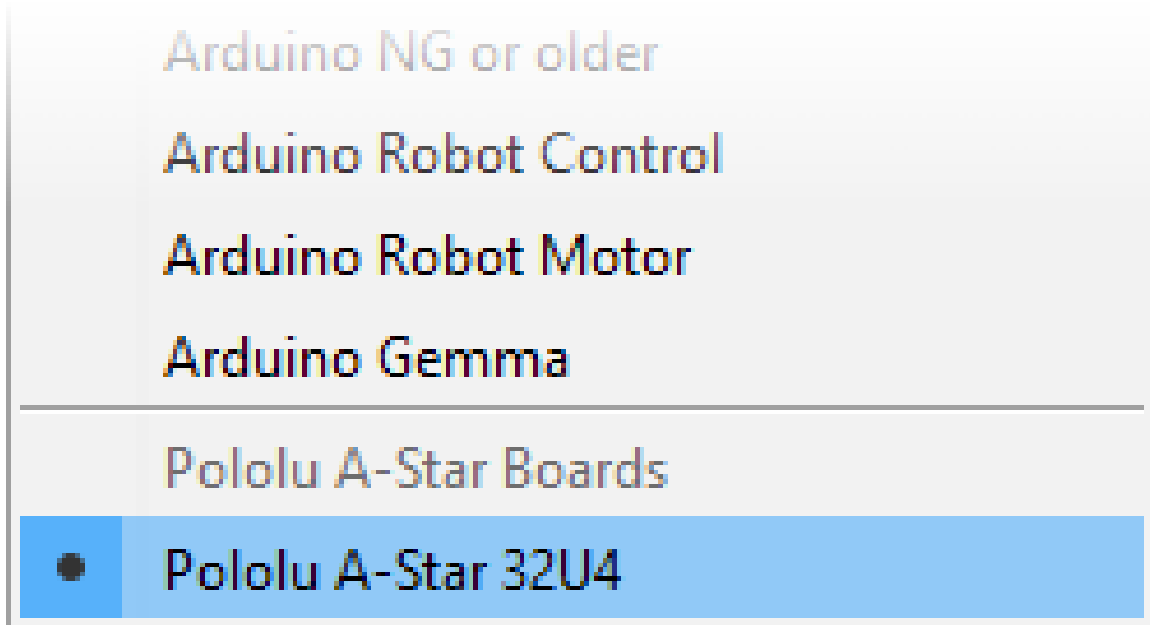
If there are already other URLs in the box, you can either add this one separated by a comma **or** click the button next to the box to open an input dialog where you can add the URL on a new line.



Adding a Boards Manager index for Pololu boards in the Arduino IDE's Preferences dialog.

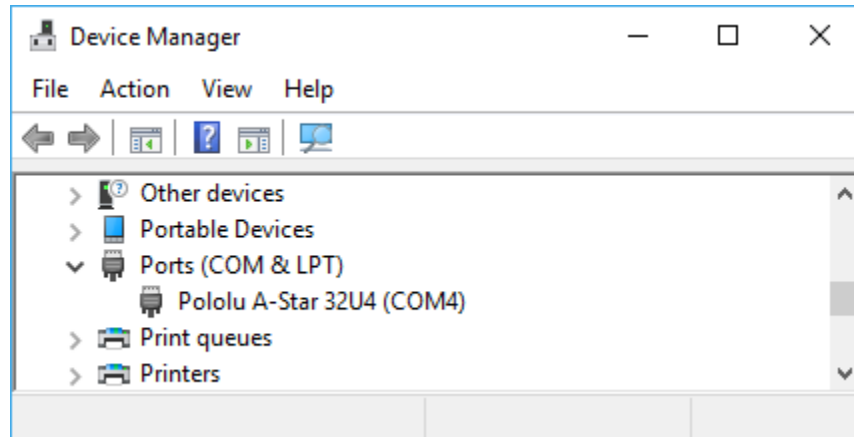
3. Click the “OK” button to close the Preferences dialog.

4. In the **Tools > Board** menu, select “Boards Manager...” (at the top of the menu).
5. In the Boards Manager dialog, search for “Pololu A-Star Boards”.
6. Select the “Pololu A-Star Boards” entry in the list, and click the “Install” button.
7. After the installation finishes, click the “Close” button to close the Boards Manager dialog.
8. In the **Tools > Board** menu, select the “Pololu A-Star 32U4” entry. If you do not see your device listed in the Board menu, try restarting the Arduino IDE.



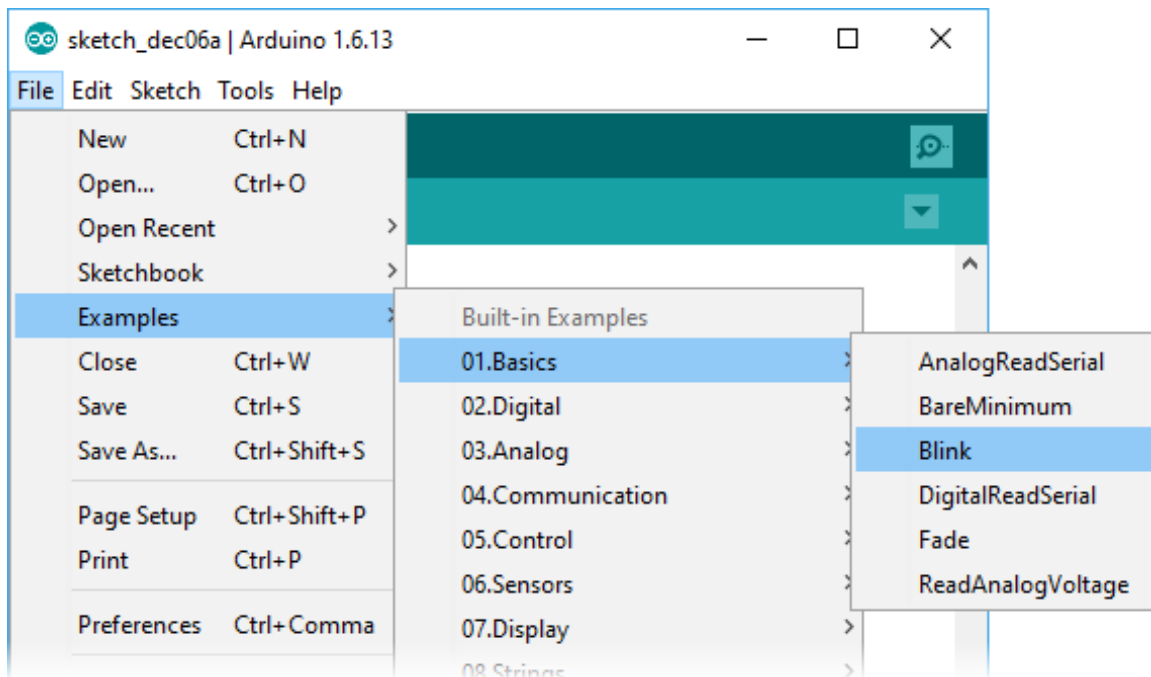
Selecting the Pololu A-Star 32U4 in the Boards menu.

9. In the **Tools > Port** menu, select the port for the device. On Windows you can determine what COM port the device is assigned to by looking at the “Ports (COM & LPT)” section of the Device Manager. On Linux, the port name will begin with “/dev/ttyACM”. On Mac OS X, the port name will begin with “/dev/tty.usbmodem”.



Windows 10 Device Manager showing the A-Star's virtual COM port.

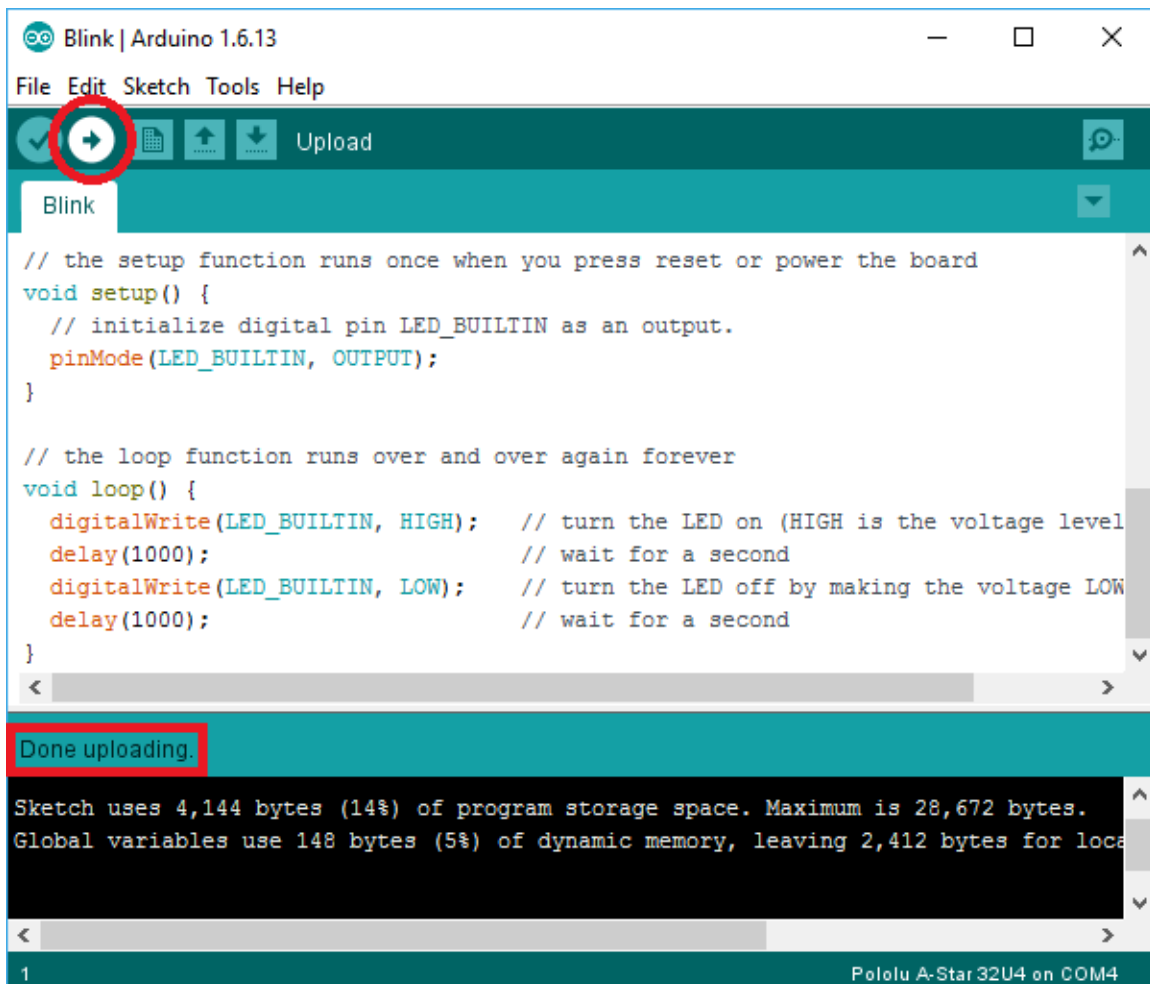
10. Open up the "Blink" Arduino example, which can be found under **File > Examples > 01.Basics > Blink**. The code in this example will blink the yellow LED. When you select the Blink example, a new Arduino IDE window will open up. It is OK to close the first window.



Selecting the Blink example in the Arduino IDE.

11. Press the "Upload" button to compile the sketch and upload it to the device. If everything goes correctly, you will see the message "Done uploading" appear near the bottom of the window. If you are using Windows and you have not previously programmed an A-Star device on this

USB port, then Windows might take several seconds to recognize the A-Star bootloader. The bootloader times out after 8 seconds and goes back to running the sketch, so the upload might fail if Windows does not recognize it quickly enough. If this happens, try again. If you are using Windows XP and have not programmed an A-Star on this USB port, you will have to go through the Found New Hardware Wizard again as described in the previous section, but the second time you try to upload it should work. If the Arduino IDE has trouble connecting to the port or using it, try unplugging the device, closing any programs that might be using the serial port, restarting the Arduino IDE, and then plugging the device back in.



Uploading a sketch to the A-Star using the Arduino IDE.

12. If you uploaded the Blink sketch, then the yellow LED should be blinking once every two seconds. However, we ship some A-Stars with that same example already programmed onto it, so you might not be convinced that anything has changed. Try changing the delay values in the sketch to something else and uploading again to see if you can change the speed of

the LED.



The A-Star 32U4 boards are similar enough to the Arduino Leonardo that you do not actually have to install the add-on. If you want to, you can just select the “Arduino Leonardo” board in the Arduino IDE. Note that if you upload a sketch to the device this way, your computer will then recognize it as a Leonardo (for example, its entry in the Windows Device Manager will display “Arduino Leonardo”).

After you succeed in programming your device from the Arduino IDE, there are many resources you can use to learn more:

- The Arduino IDE has many **examples** [<http://arduino.cc/en/Tutorial/HomePage>] that can run on A-Stars.
- The Arduino website has a **Language Reference** [<http://arduino.cc/en/Reference/HomePage>], a wiki called the **The Arduino Playground** [<http://playground.arduino.cc/>], and other resources.
- The A-Star 32U4 boards are similar to the **Arduino Leonardo** [<https://www.pololu.com/product/2192>] and **Arduino Micro** [<https://www.pololu.com/product/2188>], so you can search the Internet for relevant projects that use one of those boards.
- The Related Resources section lists many more resources.

4.3. Programming using avr-gcc and AVRDUDE

This section explains how to program our 32U4 family of boards using the avr-gcc toolchain and AVRDUDE. This section is intended for advanced users who do not want to use the Arduino IDE as described in the previous section.

Getting the prerequisites

If you are using Windows, we recommend downloading **WinAVR** [<http://winavr.sourceforge.net/>], which contains the avr-gcc toolchain and a command-line utility called **AVRDUDE** [<http://www.nongnu.org/avrdude/>] that can be used to upload programs to the A-Star bootloader. If the version of GNU Make that comes with WinAVR crashes on your computer, we recommend using the **Pololu version of GNU Make** [<https://github.com/pololu/make/releases>].

If you are using Mac OS X, we recommend downloading the **CrossPack for AVR Development** [<http://www.obdev.at/products/crosspack>].

If you are using Linux, you will need to install avr-gcc, avr-libc, and AVRDUDE. Ubuntu users can get the required software by running:

```
sudo apt-get install gcc-avr avr-libc avrdude
```

After you have installed the prerequisites, open a command prompt and try running these commands to make sure all the required utilities are available:

```
avr-gcc -v
avr-objcopy -V
make -v
avrdude
```

If any of those commands fail, make sure the desired executable is installed on your computer and make sure that it is in a directory listed in your PATH environment variable.

Compiling an example program

Copy the following code to a file named “main.c”:

```

1 | #define F_CPU 16000000
2 | #include <avr/io.h>
3 | #include <util/delay.h>
4 |
5 | int main()
6 | {
7 |     DDRC |= (1 << DDC7);    // Make pin 13 be an output.
8 |     while(1)
9 |     {
10 |         PORTC |= (1 << PORTC7); // Turn the LED on.
11 |         _delay_ms(500);
12 |         PORTC &= ~(1 << PORTC7); // Turn the LED off.
13 |         _delay_ms(500);
14 |     }
15 | }
```

In the same folder, create a file named “Makefile” with the following contents:

```

PORT=\\\\.\\USBSER000
MCU=atmega32u4
CFLAGS=-g -Wall -mcall-prologues -mmcu=$(MCU) -Os
LDFLAGS=-Wl,-gc-sections -Wl,-relax
CC=avr-gcc
TARGET=main
OBJECT_FILES=main.o

all: $(TARGET).hex

clean:
    rm -f *.o *.hex *.obj *.hex

%.hex: %.obj
    avr-objcopy -R .eeprom -O ihex $< $@

%.obj: $(OBJECT_FILES)
    $(CC) $(CFLAGS) $(OBJECT_FILES) $(LDFLAGS) -o $@

program: $(TARGET).hex
    avrdude -p $(MCU) -c avr109 -P $(PORT) -U flash:w:$(TARGET).hex
```

Make sure that the PORT variable in the Makefile is set to the name of the device's virtual serial port.

In Windows, `\\\\.\\usbser000` should work if the A-Star is the only USB device connected that is using the `usbser.sys` driver, but you can change it to be the actual name of the COM port (e.g. `COM13`).

In a command prompt, navigate to the directory with the Makefile and `main.c`. If you run the command `make` , the code should get compiled and produce a file named “main.hex”.

Programming

To program the A-Star device, you will need to get it into bootloader mode first. One way to do this is to reset the AVR twice within 750 ms. Most of the boards in our 32U4 family have a reset button that can be used to reset the board. On any of our 32U4 family of boards, a pushbutton can be connected between the GND and RST pins to serve as a reset button, or you can use a wire. Once the device is in bootloader mode, quickly run the command `make program` to program it. If you wait longer than 8 seconds, the A-Star bootloader will exit and the AVR will go back to running the user program.

5. A-Star 32U4 Arduino library

The A-Star 32U4 can be programmed from the Arduino IDE as described in the preceding sections.

To help interface with all the on-board hardware on the A-Star 32U4, we provide the **AStar32U4 library**. The **AStar32U4 library documentation** [<https://pololu.github.io/a-star-32u4-arduino-library/>] provides detailed information about the library, and the library comes with several example sketches.

If you are using version 1.6.2 or later of the Arduino software (IDE), you can use the Library Manager to install this library:

1. In the Arduino IDE, open the “Sketch” menu, select “Include Library”, then “Manage Libraries...”.
2. Search for “AStar32U4”.
3. Click the AStar32U4 entry in the list.
4. Click “Install”.

If this does not work, you can manually install the library:

1. Download the **latest release archive from GitHub** [<https://github.com/pololu/a-star-32u4-arduino-library>] and decompress it.
2. Rename the folder “a-star-32u4-arduino-library-master” to “AStar32U4”.
3. Move the “AStar32U4” folder into the “libraries” directory inside your Arduino sketchbook directory. You can view your sketchbook location by opening the “File” menu and selecting “Preferences” in the Arduino IDE. If there is not already a “libraries” folder in that location, you should make the folder yourself.
4. After installing the library, restart the Arduino IDE.

After you install the AStar32U4 library, you can learn more about it by trying the included example sketches and by reading the **AStar32U4 library documentation** [<https://pololu.github.io/a-star-32u4-arduino-library/>].

If you are using the A-Star 32U4 Robot Controller as a Raspberry Pi add-on, you might also want to make use of our **Raspberry Pi slave library for Arduino** [<https://github.com/pololu/pololu-rpi-slave-arduino-library>], which sets up the A-Star as an I²C slave and helps establish communication with a Raspberry Pi master. See **Section 3.8** for more information about the Raspberry Pi interface.

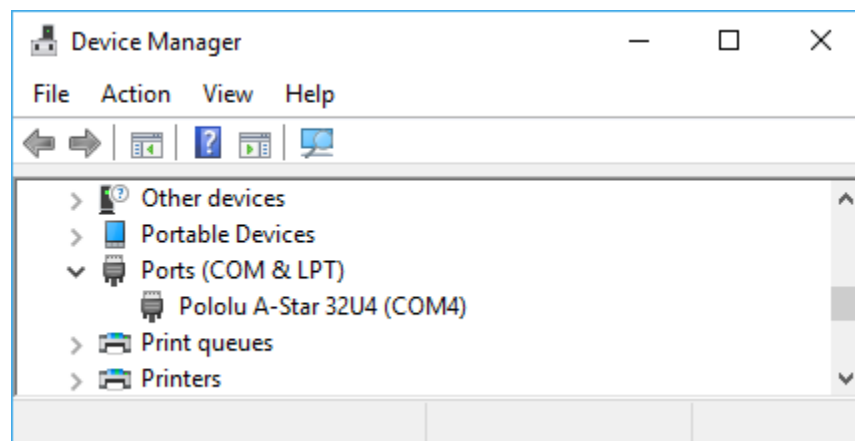
6. The A-Star 32U4 USB interface

Our 32U4 family of boards are based on a single AVR ATmega32U4 microcontroller that runs the user program and also handles the USB connection to the computer. The AVR has a full-speed USB transceiver built into it and can be programmed to present almost any type of USB device interface to the computer.

USB is an asymmetric system that consists of a single “host” connected to multiple “devices”. The host is typically a personal computer. The ATmega32U4 can only act as a USB device, so an A-Star device cannot be connected to other USB devices like mice and keyboards; it can only be connected to a host such as your computer.

Programming an ATmega32U4 board using the Arduino IDE as described earlier will automatically configure it as a composite device with a single virtual serial port. If you program the microcontroller with an Arduino sketch that implements another USB device class, like HID or MIDI, you will see additional child devices as well.

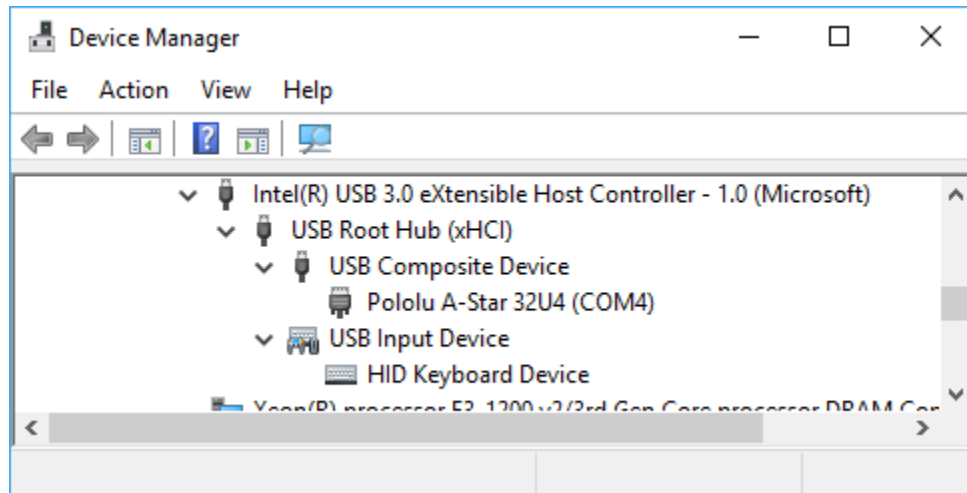
On a Windows computer, you can see the virtual serial port by going to your computer's Device Manager and expanding the “Ports (COM & LPT)” list. You should see a COM port labeled “Pololu A-Star 32U4”. In parentheses after the name, you will see the name of the port (e.g. “COM3” or “COM4”). Windows will assign a different COM port number to the device depending on what USB port you plug it into and whether it is in bootloader mode or not. If you need to change the COM port number assigned to the A-Star, you can do so using the Device Manager. Double-click on the COM port to open its properties dialog, and click the “Advanced...” button in the “Port Settings” tab. From this dialog you can change the COM port assigned to the device.



Windows 10 Device Manager showing the A-Star's virtual COM port.

On a Windows computer, you can see the rest of the USB interface by going to the Device Manager, selecting **View > Devices by connection**, and then expanding entries until you find the “Pololu A-Star

32U4" COM port. Near it, you should see the parent composite device.



The Windows 10 Device Manager in "Devices by connection" mode, showing that the A-Star is a composite device.

On a Linux computer, you can see details about the USB interface by running `lsusb -v -d 1ff8:` in a Terminal. The virtual serial port can be found by running `ls /dev/ttyACM*` in a Terminal.

On a Mac OS X computer, the virtual serial port can be found by running `ls /dev/tty.usbmodem*` in a Terminal.

You can send and receive bytes from the virtual serial port using any terminal program that supports serial ports. Some examples are the Serial Monitor in Arduino IDE, the **Pololu Serial Transmitter Utility** [<https://www.pololu.com/docs/0J23>], **Br@y Terminal** [<http://sites.google.com/site/terminalbpp/>], **PuTTY** [<http://www.chiark.greenend.org.uk/~sgtatham/putty/>], **TeraTerm** [<http://tssh2.sourceforge.jp/>], **Kermit** [<http://www.columbia.edu/kermit/ck80.html>], and **GNU Screen** [<http://www.gnu.org/software/screen/>]. Many computer programming environments also support sending and receiving bytes from a serial port.

7. The A-Star 32U4 Bootloader

Our 32U4 family of boards come with a USB bootloader that can be used in conjunction with the Arduino IDE or AVRDUDE to load new programs onto the device. This section documents some technical details of the bootloader for advanced users who want to better understand how it works. If you just want to get started using your device, it is fine to skip this section.

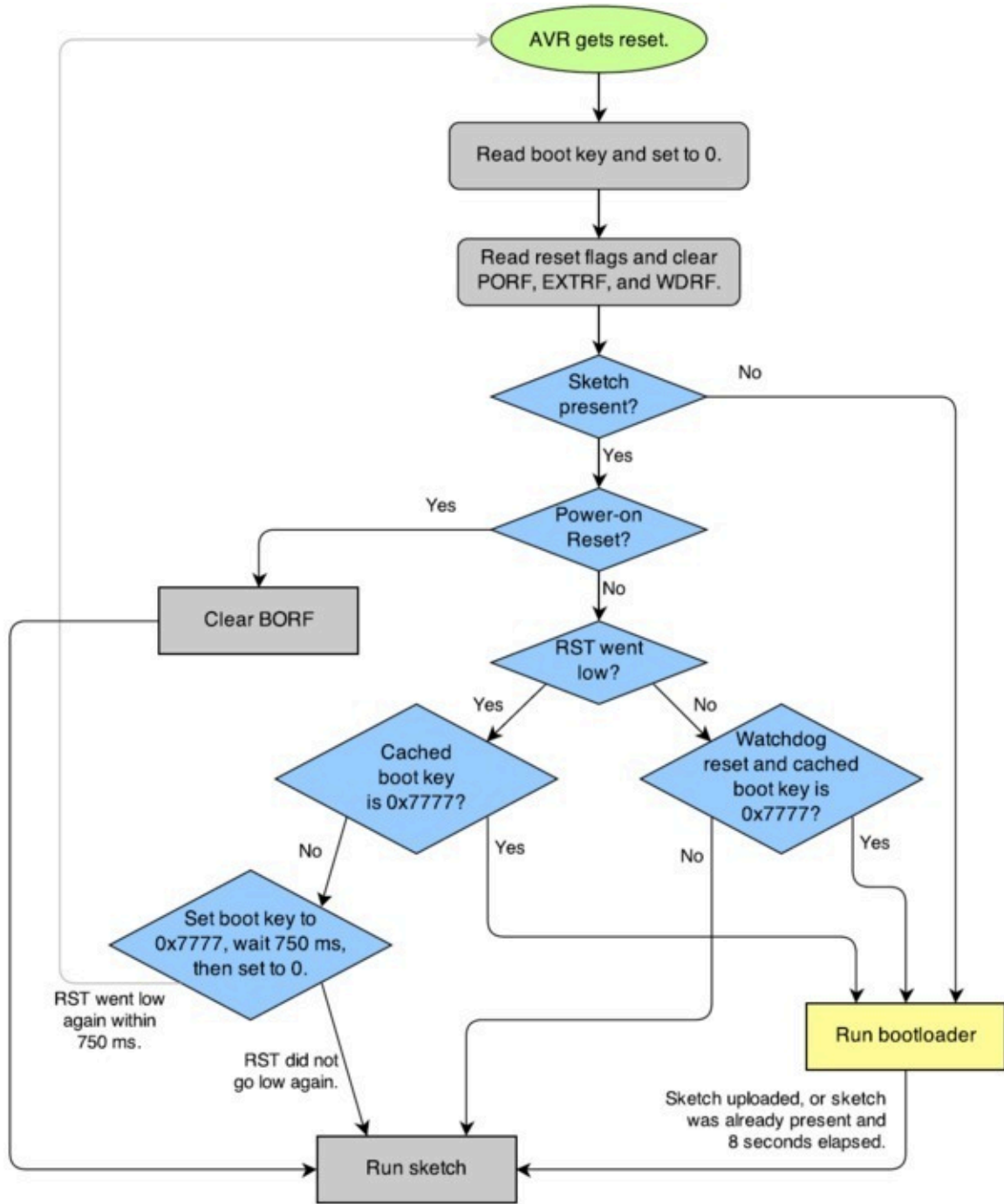
The A-Star 32U4 Bootloader is based on the **Caterina bootloader** [<https://github.com/arduino/Arduino/tree/master/hardware/arduino/avr/bootloaders/caterina>], which is the bootloader used on the **Arduino Leonardo** [<https://www.pololu.com/product/2192>], **Arduino Micro** [<https://www.pololu.com/product/2188>] and several other ATmega32U4 boards. The bootloader is open source and its **source code** [<https://github.com/pololu/a-star/tree/master/bootloaders/caterina>] is available on GitHub. The bootloader occupies the upper four kilobytes of the ATmega32U4's program memory, leaving 28 KB for the user program. The bootloader's USB interface consists of a single virtual serial port that accepts the programming commands defined in **AVR109** [<http://www.atmel.com/images/doc1644.pdf>]. The bootloader always runs first immediately after the AVR is reset.

Startup logic

The main difference between the A-Star 32U4 Bootloader and Caterina is in the startup logic. This is the part of the bootloader that runs immediately after the AVR is reset, and it decides whether to run the user program or run the rest of the bootloader. The startup logic of the Caterina bootloader is designed so that when the RST line goes low, the bootloader will run. This means that if you want to restart your program using the RST line, it will take 8 seconds before the bootloader times out waiting for an upload and the sketch starts.

The A-Star 32U4 Bootloader has different startup logic that allows you to use the RST line to reset the board with a smaller delay. If the RST line goes low once, the user program will run after a 750 ms delay. If the RST line goes low twice within 750 ms, then the bootloader will run. (This behavior is the same as on boards like SparkFun's Pro Micro.)

The start-up logic of the A-Star 32U4 Bootloader is shown in the flowchart below:



The startup logic for the A-Star 32U4 bootloader.

Brown-out detection

Unlike many other ATmega32U4 boards, our 32U4 family of boards have brown-out detection enabled. The brown-out threshold is 4.3 V, and if the voltage on VCC goes below this then the AVR will reset.

The bootloader was designed so that the user program can detect brown-out resets. To do so, check to see if the BORF bit in the MCUSR register is set, and then clear it later. Here is some example code you could put in your `setup` function for detecting brown-out resets:

```
1  pinMode(13, OUTPUT);
2  if (MCUSR & (1 << BORF))
3  {
4      // A brownout reset occurred. Blink the LED
5      // quickly for 2 seconds.
6      for(uint8_t i = 0; i < 10; i++)
7      {
8          digitalWrite(13, HIGH);
9          delay(100);
10         digitalWrite(13, LOW);
11         delay(100);
12     }
13 }
14 MCUSR = 0;
```

8. Reviving an unresponsive A-Star

In order to load a new program onto your A-Star 32U4 device, you will need to get it into bootloader mode and send programming commands to it over its virtual serial port using appropriate software. If you are programming the device from the Arduino IDE, the sketch loaded onto the device will generally support a special USB command for putting it in bootloader mode, and the Arduino IDE sends that command automatically when you click the Upload button. However, you might find yourself in a situation where the device is unresponsive and that method will not work. This can happen for two reasons:

- You accidentally loaded a malfunctioning program onto the device that is incapable of responding to the special USB command. For example, your program might be stuck in an infinite loop with interrupts disabled.
- You loaded a program which uses a non-standard type of USB interface or no USB interface.

The following sections provide different procedures you can use to revive your device.

8.1. Reviving using the Arduino IDE

This section explains two special methods for programming an A-Star (or another of our 32U4 family of boards) using the Arduino IDE in case your usual method of programming is not working. These instructions were developed for the Arduino IDE versions 1.0.5-r2 and 1.6.0, and they might need to be modified for future versions.

Reset button

If you have an A-Star 32U4 Micro, you should connect a **momentary pushbutton** [<https://www.pololu.com/product/1400>] between the GND and RST pins to serve as a reset button. Other boards in our 32U4 family have a reset button you can use. Alternatively, you can use a wire to temporarily connect GND and RST together instead of using a reset button.

Resetting the board twice within 750 ms makes the board go into bootloader mode. The bootloader will exit after 8 seconds and try to run the sketch again if it has not started receiving programming commands. To revive the device, you need to make sure you start sending it programming commands before the 8-second period is over.

In bootloader mode, the yellow LED (the one labeled *LED 13*) fades in and out. It is useful to look at this LED so you can know what mode the microcontroller is in. Also, we recommend enabling verbose output during upload using the Arduino IDE's "Preferences" dialog. Looking at the LED and looking at the verbose output during the following procedures will help you understand what is going on.

The uploading-before-bootloader method

The goal of the uploading-before-bootloader method is to select a non-existent serial port in the Arduino IDE and then make sure the Arduino IDE enters the uploading phase before the microcontroller goes into bootloader mode. This method has been tested on Arduino 1.0.5-r2 and 1.6.0. This method does not work on Arduino 1.5.6-r2 because that version of the IDE gives a fatal error message if the selected serial port is not present at the beginning of the uploading phase (e.g. “Board at COM7 is not available.”).

1. Connect the device to your computer via USB.
2. In the “Tools” menu, open the “Board” sub-menu, and select “Pololu A-Star 32U4”.
3. In the “Tools” menu, open the “Port” sub-menu, and check to see if any ports are selected. If the “Port” menu is grayed out or no ports in it are selected, that is good, and you can skip to step 6.
4. Reset the board twice to get the board into bootloader mode. While the board is in bootloader mode, quickly select the new serial port that corresponds to the bootloader in the “Port” menu.
5. After 8 seconds, the bootloader will exit and attempt to run the sketch again. Wait for the bootloader to exit. Verify that either the “Port” menu is grayed out or no ports in it are selected.
6. Click the Upload button. The Arduino IDE will compile your sketch and start uploading it.
7. As soon as the large status bar near the bottom of the IDE says “Uploading...”, reset the board twice to get into bootloader mode.

The Arduino IDE will stay in the uploading phase for 10 seconds, waiting for a new serial port to appear. Once the serial port of the bootloader appears, the Arduino IDE will connect to it and send programming commands.

The bootloader-before-uploading method

The goal of the bootloader-before-uploading method is to select the bootloader’s virtual serial port in the Arduino IDE and then make sure the board is in bootloader mode at the time when the Arduino IDE enters the uploading phase.

1. Connect the device to your computer via USB.
2. In the “Tools” menu, open the “Board” sub-menu and check to see if the “Pololu A-Star 32U4 (bootloader port)” entry is visible. If this entry is visible, you can skip to step 6.
3. If you are using a 1.0.x version of the Arduino IDE, open the file **[sketchbook location]/hardware/pololu/boards.txt** using a text editor. If you are using a 1.5.x version of the Arduino IDE, open the file **[sketchbook location]/hardware/pololu/avr/boards.txt**

using a text editor. You can see the sketchbook location in the Arduino IDE preferences dialog. The file you are looking for is part of the A-Star add-on.

4. In the `boards.txt` file that you opened, find the lines at the bottom of the file that start with `#a-star32U4bp` . Uncomment each of those lines by deleting the “#” character, and then save the file.
5. Close the Arduino IDE and restart it.
6. In the “Tools” menu, open the “Board” sub-menu and select “Pololu A-Star 32U4 (bootloader port)”. This entry is configured so that the Arduino IDE will send programming commands directly to selected serial port, instead of trying to send a special USB command to the port to get it into bootloader mode and then waiting for the new port to appear. By selecting this entry, the timing of the programming process below becomes easier, especially on Windows.
7. Prepare the computer to show you a list of its virtual serial ports. If you are using Windows, this means you should open the Device Manager. If you are on Linux or Mac OS X, this means you should open a Terminal and type the command `ls /dev/tty*` but do not press enter until the board is in bootloader mode in the next step.
8. Reset the board twice to get the board into bootloader mode. While it is in bootloader mode, quickly look at the list of serial ports provided by your operating system in order to determine what port the bootloader is assigned to.
9. Reset the board twice to get the board into bootloader mode again. While the board is in bootloader mode, quickly select the serial port of the bootloader in the Arduino IDE. The port can be selected in the “Port” sub-menu under “Tools”.
10. In the Arduino IDE, click the “Verify” button to compile your sketch. This could make the timing easier during the next step.
11. Press the reset button twice to get the board into bootloader mode again. As soon as you see the yellow LED fading in and out, press the Upload button.

The Arduino IDE will compile your sketch and then upload it to the selected serial port.

If the compilation of the sketch takes longer than 8 seconds, then this procedure will fail because the bootloader will time out and start trying to run the malfunctioning sketch again. If that happens, try the procedure again using a simpler sketch such as the Blink example that can be found under **File > Examples > 01.Basics > Blink**.

After reviving your device, be sure to change the Board setting back to “Pololu A-Star 32U4” and select the right Port.

8.2. Reviving using AVRDUDE

This section explains a special method for reviving an A-Star (or another of our 32U4 family of boards)

using the command-line utility **AVRDUDE** [<http://www.nongnu.org/avrdude/>] in case your usual method of programming is not working. AVRDUDE stands for “AVR Downloader/UplodaDEr”, and it is compatible with the A-Star bootloader.

If you have an A-Star 32U4 Micro, you should connect a **momentary pushbutton** [<https://www.pololu.com/product/1400>] between the GND and RST pins to serve as a reset button. Other boards in our 32U4 family have a reset button you can use. Alternatively, you can use a wire to temporarily connect GND and RST together instead of using a reset button.

1. Connect the device to your computer via USB.
2. Prepare the computer to show you a list of its virtual serial ports. If you are using Windows, this means you should open the Device Manager. If you are on Linux or Mac OS X, this means you should open a Terminal and type the command `ls /dev/tty*` but do not press enter until the board is in bootloader mode in the next step.
3. Press the reset button twice within 750 ms to make the AVR go into bootloader mode. You should see the yellow LED fading in and out when the AVR is in bootloader mode. While it is in bootloader mode, quickly look at the list of serial ports provided by your operating system in order to determine what port the bootloader is assigned to.
4. Type the following command in your terminal and replace COM4 with the name of the bootloader's serial port, but do not press enter yet. This command will erase the malfunctioning program on the device but preserve the bootloader.

```
avrdude -c avr109 -p atmega32U4 -P COM4 -e
```
5. Press the reset button twice within 750 ms to make the AVR go into bootloader mode.
6. Quickly run the command you typed previously. The command needs to be run within 8 seconds of starting the bootloader, or else the bootloader will exit and try to run the malfunctioning program again.

By following the instructions above, the malfunctioning program on the device will be erased and the device will stay in bootloader mode indefinitely. You can now load another program onto it using the Arduino IDE or AVRDUDE.

9. Related resources

To learn more about using the Pololu A-Star boards, see the following list of resources:

- The Arduino IDE has many **examples** [<http://arduino.cc/en/Tutorial/HomePage>] that can run on the A-Stars.
- The Arduino website has a **Language Reference** [<http://arduino.cc/en/Reference/HomePage>], a wiki called the **The Arduino Playground** [<http://playground.arduino.cc/>], and other resources.
- The A-Star boards are similar to the **Arduino Leonardo** [<https://www.pololu.com/product/2192>] and **Arduino Micro** [<https://www.pololu.com/product/2188>], so you can search the Internet for relevant projects that use one of those boards.
- **Atmel's ATmega32U4 documentation** [<https://www.microchip.com/wwwproducts/en/ATmega32u4>] has the ATmega32U4 datasheet and many related documents.
- **AVR Libc Home Page** [<http://www.nongnu.org/avr-libc/>]: this page documents the standard library of functions that you can use with GNU C and C++ compilers for the AVR.
- **A-Star 32U4 Arduino library** [<https://github.com/pololu/a-star-32u4-arduino-library>]
- **AStar32U4 library documentation** [<https://pololu.github.io/a-star-32u4-arduino-library/>]
- **LUFA – the Lightweight USB Framework for AVRs** [<http://www.fourwalledcubicle.com/LUFA.php>]
- **WinAVR** [<http://winavr.sourceforge.net/>]
- **Atmel Studio 7** [<https://www.microchip.com/avr-support/atmel-studio-7>]
- **AVRDUDE** [<http://www.nongnu.org/avrdude/>]
- **AVR Freaks** [<http://www.avrfreaks.net/>]

Datasheets for some of the components found on the A-Star 32U4 boards are available below (not all components apply to every board):

- **Texas Instruments TPS2113A power multiplexer datasheet** [<https://www.pololu.com/file/0J771/tps2113a.pdf>] (1MB pdf)
- **Texas Instruments DRV8838 motor driver datasheet** [<https://www.pololu.com/file/0J806/drv8838.pdf>] (1MB pdf)
- **Maxim MAX14870 motor driver datasheet** [<https://www.pololu.com/file/0J885/MAX14870.pdf>] (492k pdf)

Finally, we would like to hear your comments and questions on the **A-Star section of the Pololu Robotics Forum** [<http://forum.pololu.com/viewforum.php?f=10>]!