

Measuring battery voltage on Arduino Uno board

This documentation is supplied with the Arduino SDK example code `heart_rate_example_with_battery_service`.

A battery shield compatible with the Arduino Uno board is available at battery-shield.com.

The method of measuring the battery voltage on an arduino board is to connect a voltage divider like shown in Figure 1. The ADC reference voltage in the example is set to 1.1V so the voltage range exposed to analog input pin 0 (A0) should be 0-1.1V.

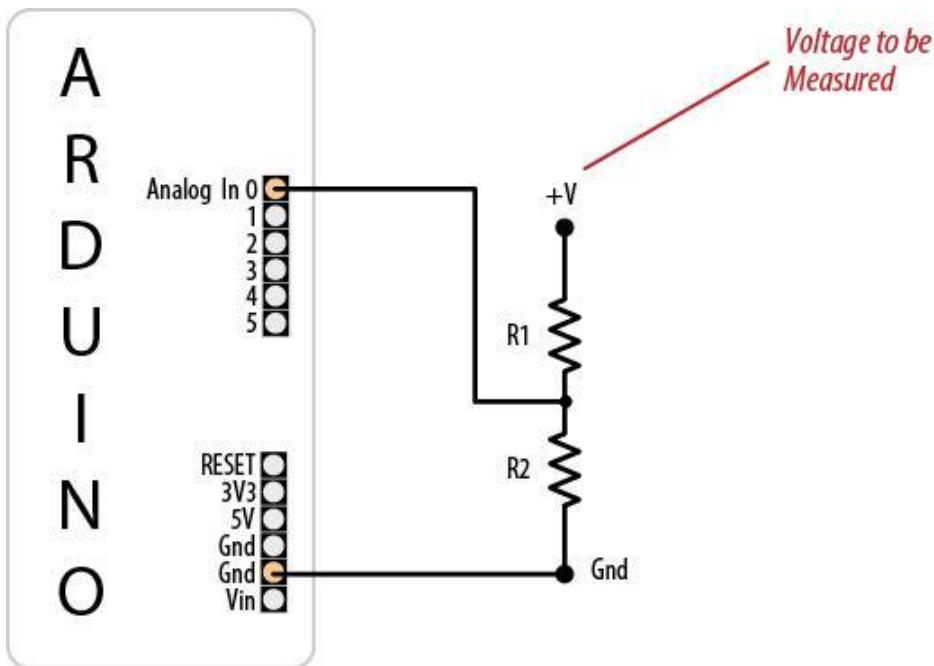


Figure 1. Setup of voltage divider for the Arduino Uno

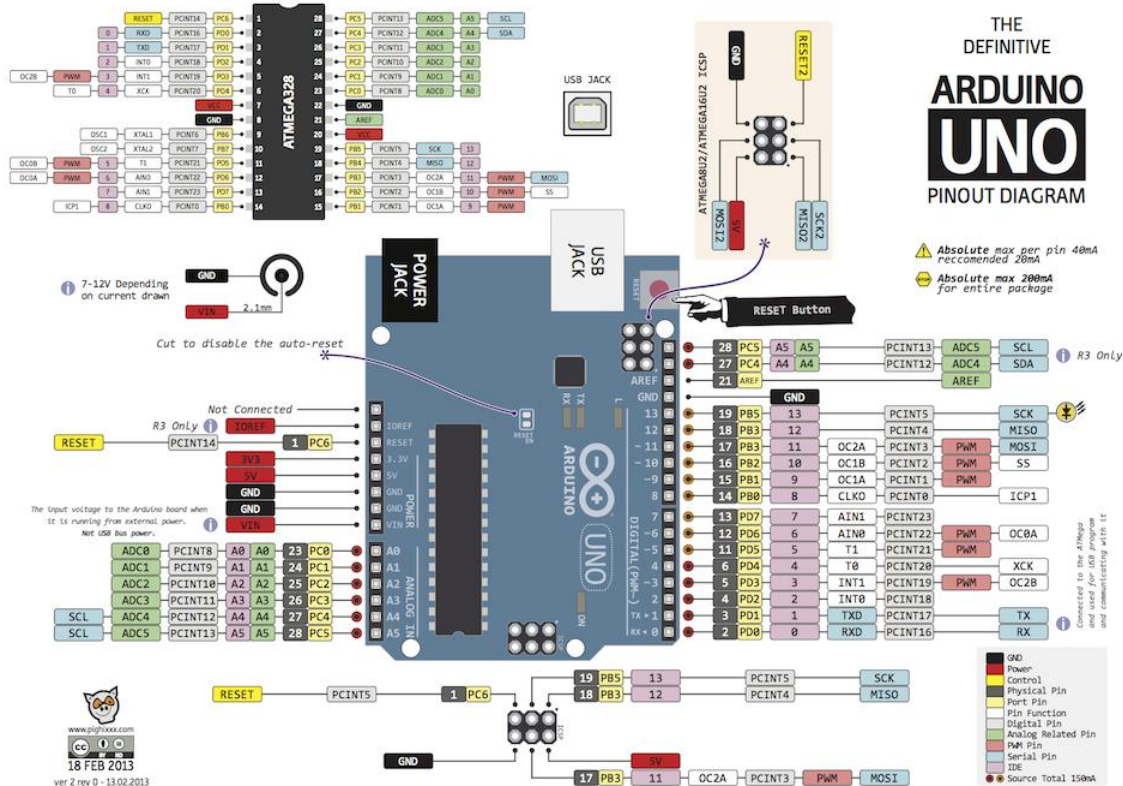


Figure 2. Arduino Uno pinout diagram

The VIN pin on the Arduino Uno board should be connected to the voltage divider as this pin represents the battery voltage.

The battery shield used provides 9 volts so the voltage divider is calibrated in a way that A0 voltage is close to 1.1V when battery voltage is 9V. In code, we will choose the 1.1V band gap voltage as the ADC reference, which will make the ADC have a voltage range of the ADC to be 0V-1.1V. This means that if the input voltage is 0V the ADC will output 0 and if the input voltage is 1.1V the ADC will output value 1023. Similarly, if the input voltage is 0.5V the ADC should output $0.5V/1.1V \times 1023 = 465$.

For the ADC, it is recommended in the AtMega328 data sheet to set impedance seen by the analog input (A0, A1, A2, A3, A4 or A5) to be maximum 10k in order to obtain accurate enough ADC reading. However, the data sheet also states that higher impedance values are allowed for slow sampling rates. If current consumption of the device is of any importance, we would like to have higher resistance than 10k in order to save current. When choosing to have high resistance values, it is advised to connect a capacitor as well between the Arduino analog input pin and ground.

Example of dimensioning voltage divider for the ADC

Lets dimension R1 and R2 in a way that the max battery voltage will give us 1.05V at the AIN analog input pin. To choose 1.05V volts instead of 1.1V will give room for calibration later on. This way, we could also measure battery voltage slightly over 9V

without having the ADC to saturate. Lets choose R2 as 1Mohm as that resistance will enable relatively low current leakage through the voltage divider. We then calculate the value for R2 with the equation:

$$R_1 = (V_{\text{batt}} * R_2 / V_{\text{AIN}}) - R_2 \quad (\text{equation 1})$$

Where we input

$$V_{\text{batt}} = 9.00 \text{ V}$$

$$R_2 = 1 \text{ M}\Omega$$

$$V_{\text{AIN}} = 1.05 \text{ V}$$

Which gives us $R_2 = 7.57 \text{ M}\Omega$

However, it might not be actual to have available exactly this resistor value of 7.57 MΩ. Lets assume that you will find resistors in your collection of R1=820kΩ and two others that have values of 18MΩ and 10MΩ. When connecting them in parallell, that should give R2=6.43MΩ. However, when measuring the resistors it shows that the actual values are R1=822kΩ and R2=6.49MΩ.

The resistor values are not exactly what we wanted so we need to calculate what voltage will be present at VAIN at our maximum supply voltage of 9.00V. We most likely need to calculate this because it is really hard to measure, unless you have available very high impedance voltage meter. A typical voltage meter measures voltage through 1MΩ-10MΩ resistor, which would give incorrect measurement of the AIN voltage since it is connected in parallell with R2=1MΩ. To calculate VAIN we use equation 1 and isolate for VAIN and get VAIN=1.0118V. This gives an expected maximum ADC value of $1.0118\text{V}/1.1\text{V} * 1023 = 941$ when the battery voltage is 9.00V

When the resistance of the voltage divider is so great, it apparently creates an ADC output error and unstability. Coupling a 22nF capacitor from AIN to ground will correct this error and make the ADC relatively stable. This voltage divider has been tested with sampling frequency of up to 130Hz.

The current leakage of this voltage divider is $9\text{V}/(6490\text{k}\Omega + 822\text{k}\Omega) = 1.2\mu\text{A}$

The battery monitoring software

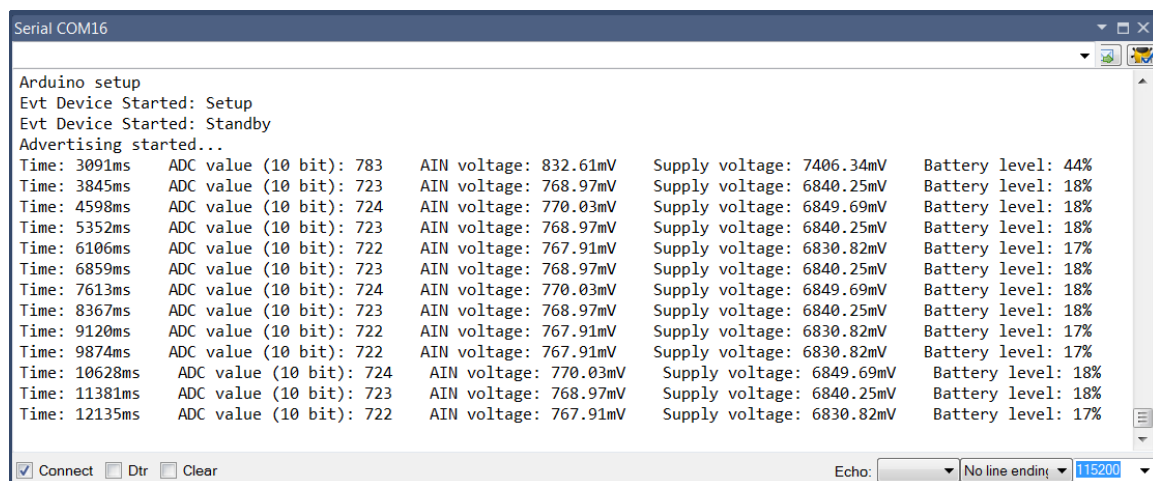
The heart rate monitor example in the SDK includes battery monitoring. The battery monitor function is constructed as follows.

- At the top of the .ino file, there is a timer setup in function Timer1start(). The value of the timer prescaler in register TCCR1B is set to 0x05 which generates timer interrupt once per second. Change the TCCR1B value for different timer interrupt frequency.

- The interrupt handler is implemented in `ISR(TIMER1_OVF_vect)` which triggers both the heart rate simulation and the battery measurement. The heart rate simulation is executed on every run of the interrupt handler but the battery measurement can have lower frequency by changing the value of `BATT_MEASUREMENT_DELAY` constant.
- In the `measure_battery(aci_state_t *aci_state)` function, voltage is measured on an analog input pin after selecting the 1.1V internal reference voltage for the ADC. The output is a 10 bit number. With a transform formula, the ADC output value is transformed into millivolts which should represent the millivolts on the ADC AIN input pin. However, there may be offset and/or gain error present, which can be corrected with adding an `OFFSET_ERROR_CALIBRATION_FACTOR` to the ADC output value to correct any observed offset error and/or multiply with `GAIN_ERROR_CALIBRATION_FACTOR` to correct any observed gain error. To fully understand the concept of offset and gain errors, look at the doc8161 AtMega328 datasheet page 259. To understand how to calibrate the ADC in order to correct any offset and/or gain error look at document ‘Characterization and Calibration of the ADC on an AVR’.
- The `measure_battery` function calculates the supply voltage and feeds that as an input to the `lib_battery_level_percent_alkaline_9_volt(float mvolts)` function which calculates the battery level as percentage, but it is only suitable for alkaline batteries.

Evaluating ADC measurement accuracy

To evaluate the correctness of the ADC output, enable battery measurement printout by setting the `ENABLE_BATTERY_MEASUREMENT_PRINTOUT` constant to 1 in the .ino file. This will generate printout on the serial port similar to the one shown in Figure 3. The ‘ADC value’ is the output from the ADC. The ‘AIN voltage’, ‘Supply voltage’ and ‘Battery voltage’ are calculated values based on the ‘ADC value’ and the `R1_RESISTANCE_KOHM` and `R2_RESISTANCE_KOHM` values.



```

Serial COM16
-----
Arduino setup
Evt Device Started: Setup
Evt Device Started: Standby
Advertising started...
Time: 3091ms   ADC value (10 bit): 783   AIN voltage: 832.61mV   Supply voltage: 7406.34mV   Battery level: 44%
Time: 3845ms   ADC value (10 bit): 723   AIN voltage: 768.97mV   Supply voltage: 6840.25mV   Battery level: 18%
Time: 4598ms   ADC value (10 bit): 724   AIN voltage: 770.03mV   Supply voltage: 6849.69mV   Battery level: 18%
Time: 5352ms   ADC value (10 bit): 723   AIN voltage: 768.97mV   Supply voltage: 6840.25mV   Battery level: 18%
Time: 6106ms   ADC value (10 bit): 722   AIN voltage: 767.91mV   Supply voltage: 6830.82mV   Battery level: 17%
Time: 6859ms   ADC value (10 bit): 723   AIN voltage: 768.97mV   Supply voltage: 6840.25mV   Battery level: 18%
Time: 7613ms   ADC value (10 bit): 724   AIN voltage: 770.03mV   Supply voltage: 6849.69mV   Battery level: 18%
Time: 8367ms   ADC value (10 bit): 723   AIN voltage: 768.97mV   Supply voltage: 6840.25mV   Battery level: 18%
Time: 9120ms   ADC value (10 bit): 722   AIN voltage: 767.91mV   Supply voltage: 6830.82mV   Battery level: 17%
Time: 9874ms   ADC value (10 bit): 722   AIN voltage: 767.91mV   Supply voltage: 6830.82mV   Battery level: 17%
Time: 10628ms  ADC value (10 bit): 724   AIN voltage: 770.03mV   Supply voltage: 6849.69mV   Battery level: 18%
Time: 11381ms  ADC value (10 bit): 723   AIN voltage: 768.97mV   Supply voltage: 6840.25mV   Battery level: 18%
Time: 12135ms  ADC value (10 bit): 722   AIN voltage: 767.91mV   Supply voltage: 6830.82mV   Battery level: 17%

```

Figure 3. Battery measurement printout in the Heart Rate Monitor example

Then follow these steps in order to evaluate the ADC output accuracy and correct potential ADC error:

- Use a power supply to provide a stable known voltage to the voltage divider.
- Measure the actual resistance values of R1 and R2 in the voltage divider, insert those values into the `R1_RESISTANCE_KOHM` and `R2_RESISTANCE_KOHM` constants in the `battery.cpp` file.
- Run the heart rate example with the battery measurement printout enabled and compare the measured supply voltage with the actual supply voltage.
- If the ADC output value does not match with your `V_AIN` voltage then there is offset and/or gain error present. Calibrate the ADC output by changing the `OFFSET_ERROR_CALIBRATION_FACTOR` and/or the `GAIN_ERROR_CALIBRATION_FACTOR` until you get an acceptable accuracy for the whole voltage range of your supply.

When correct AIN voltage has been measured and calculated, function `lib_battery_level_percent_alkaline_9_volt(float mvolts)` is called which outputs the battery level as a percentage. Note that the function outputs an approximation of the battery level given that the battery package is 6x1.5V alkaline batteries.