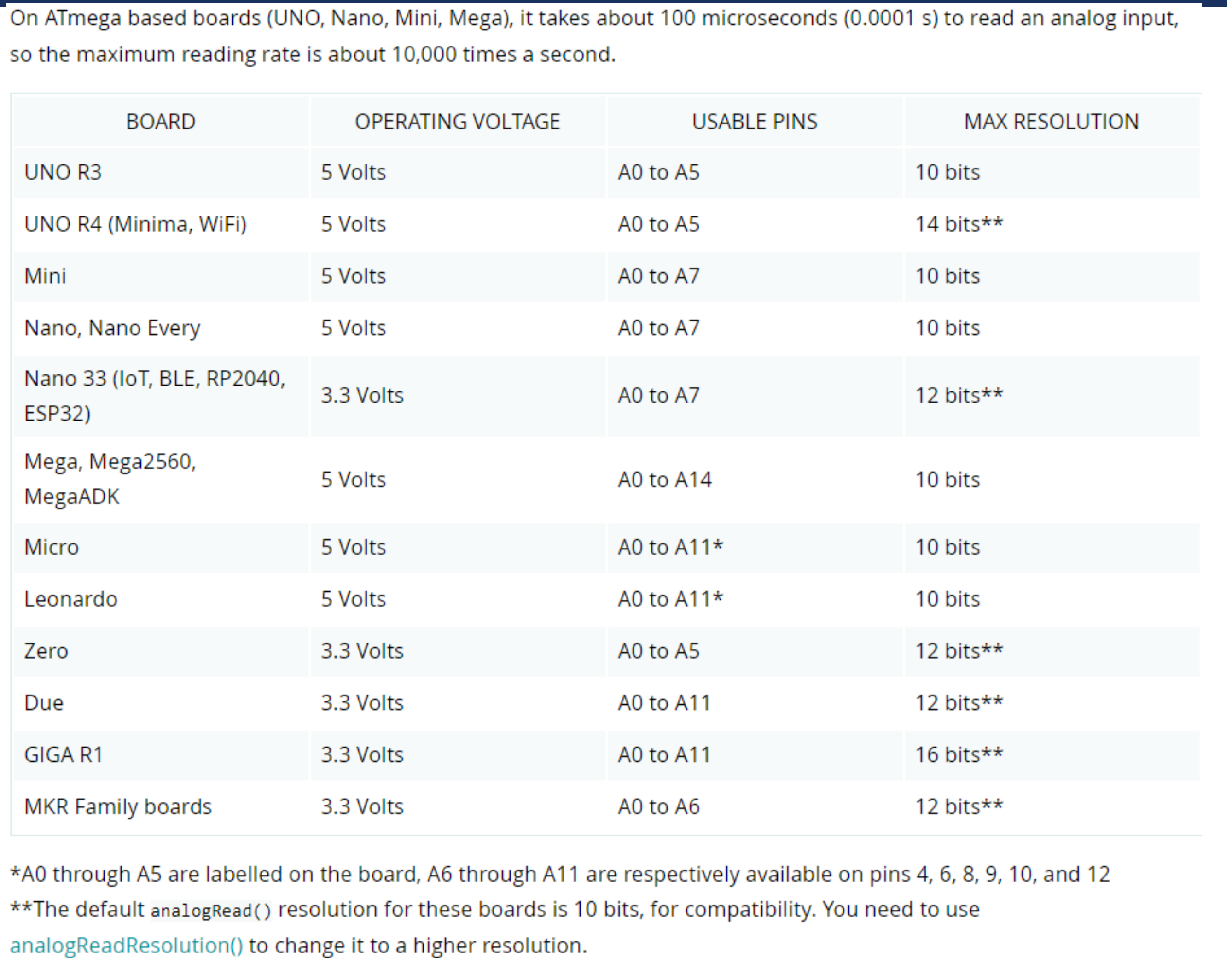
Microcontrollers & Applications

Lecture 3.3: Analog Input & PWM Output

Arduino: Analog Inputs

- analogRead(pin)
  - pin: An where n = [0,5] in general
- analogReadResolution(bits)
  - 8, 10, 12, 16 bits if available



On ATmega based boards (UNO, Nano, Mini, Mega), it takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

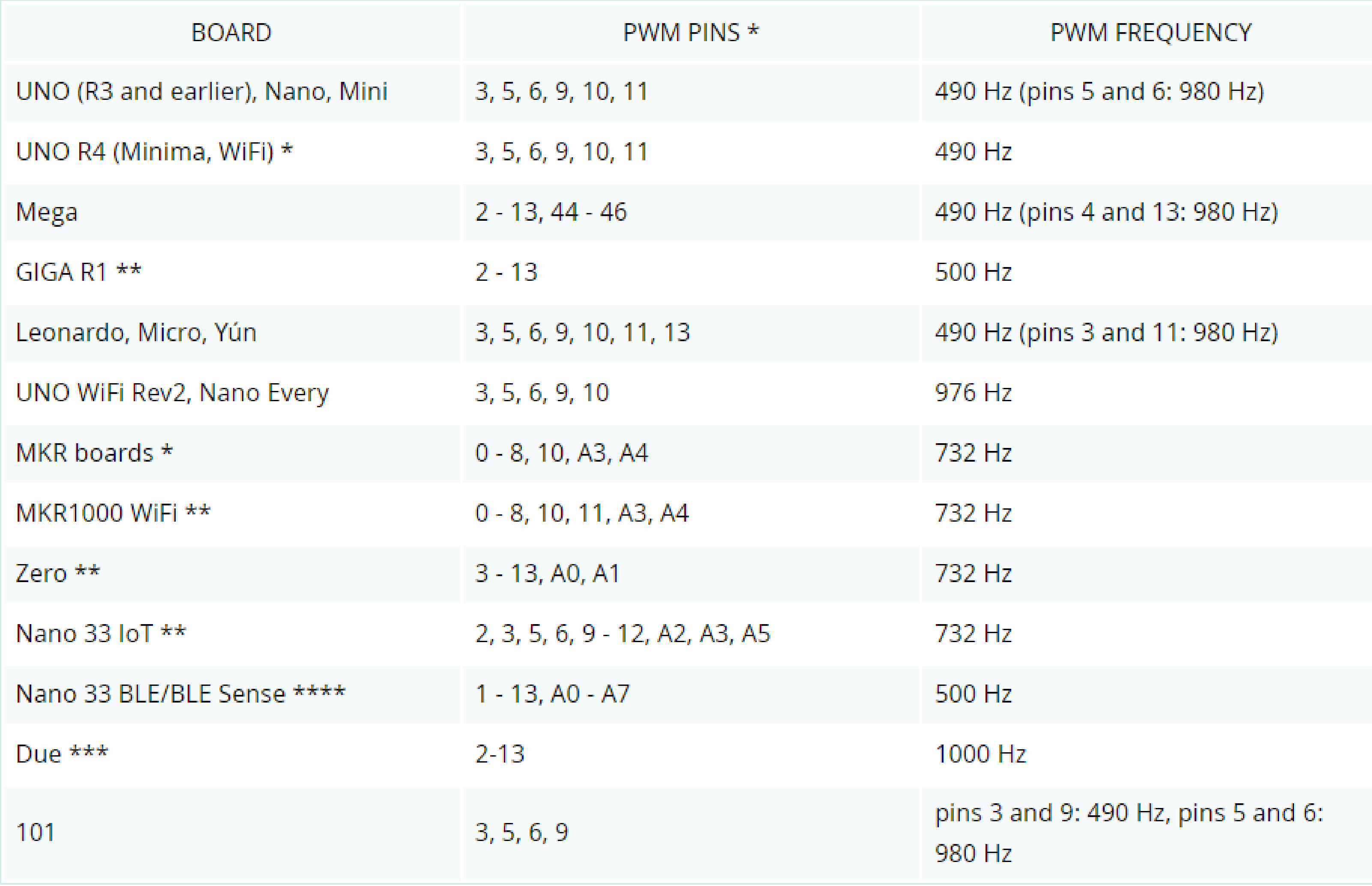
BOARD	OPERATING VOLTAGE	USABLE PINS	MAX RESOLUTION
UNO R3	5 Volts	A0 to A5	10 bits
UNO R4 (Minima, WiFi)	5 Volts	A0 to A5	14 bits**
Mini	5 Volts	A0 to A7	10 bits
Nano, Nano Every	5 Volts	A0 to A7	10 bits
Nano 33 (IoT, BLE, RP2040, ESP32)	3.3 Volts	A0 to A7	12 bits**
Mega, Mega2560, MegaADK	5 Volts	A0 to A14	10 bits
Micro	5 Volts	A0 to A11*	10 bits
Leonardo	5 Volts	A0 to A11*	10 bits
Zero	3.3 Volts	A0 to A5	12 bits**
Due	3.3 Volts	A0 to A11	12 bits**
GIGA R1	3.3 Volts	A0 to A11	16 bits**
MKR Family boards	3.3 Volts	A0 to A6	12 bits**

*A0 through A5 are labelled on the board, A6 through A11 are respectively available on pins 4, 6, 8, 9, 10, and 12

**The default `analogRead()` resolution for these boards is 10 bits, for compatibility. You need to use analogReadResolution() to change it to a higher resolution.

Arduino: PWM Outputs (1)

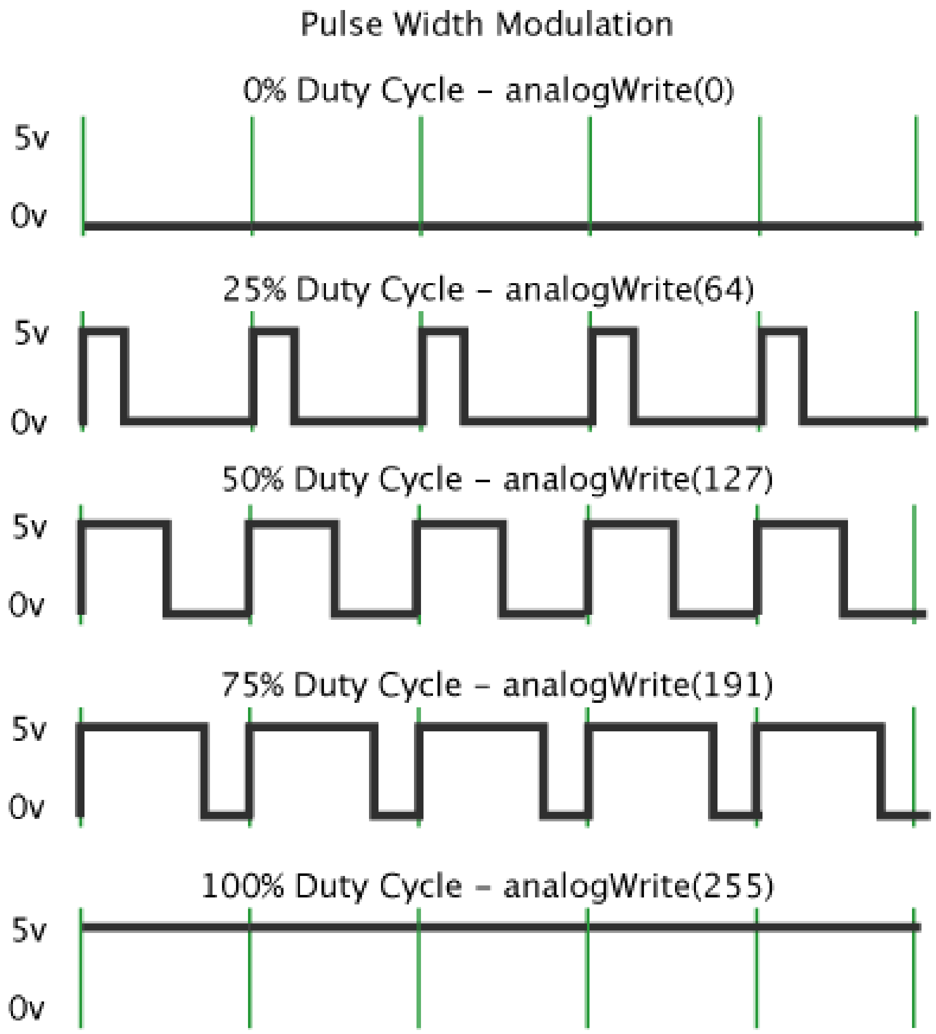
- analogWrite(pin, value)
  - pin: An where n = [0,5] in general
  - value: the duty cycle between 0 (always off) and 255 (always on)
- By changing the value, we change the power ratio transferred to the device to control
  - dc motor speed,
  - led brightness, etc.



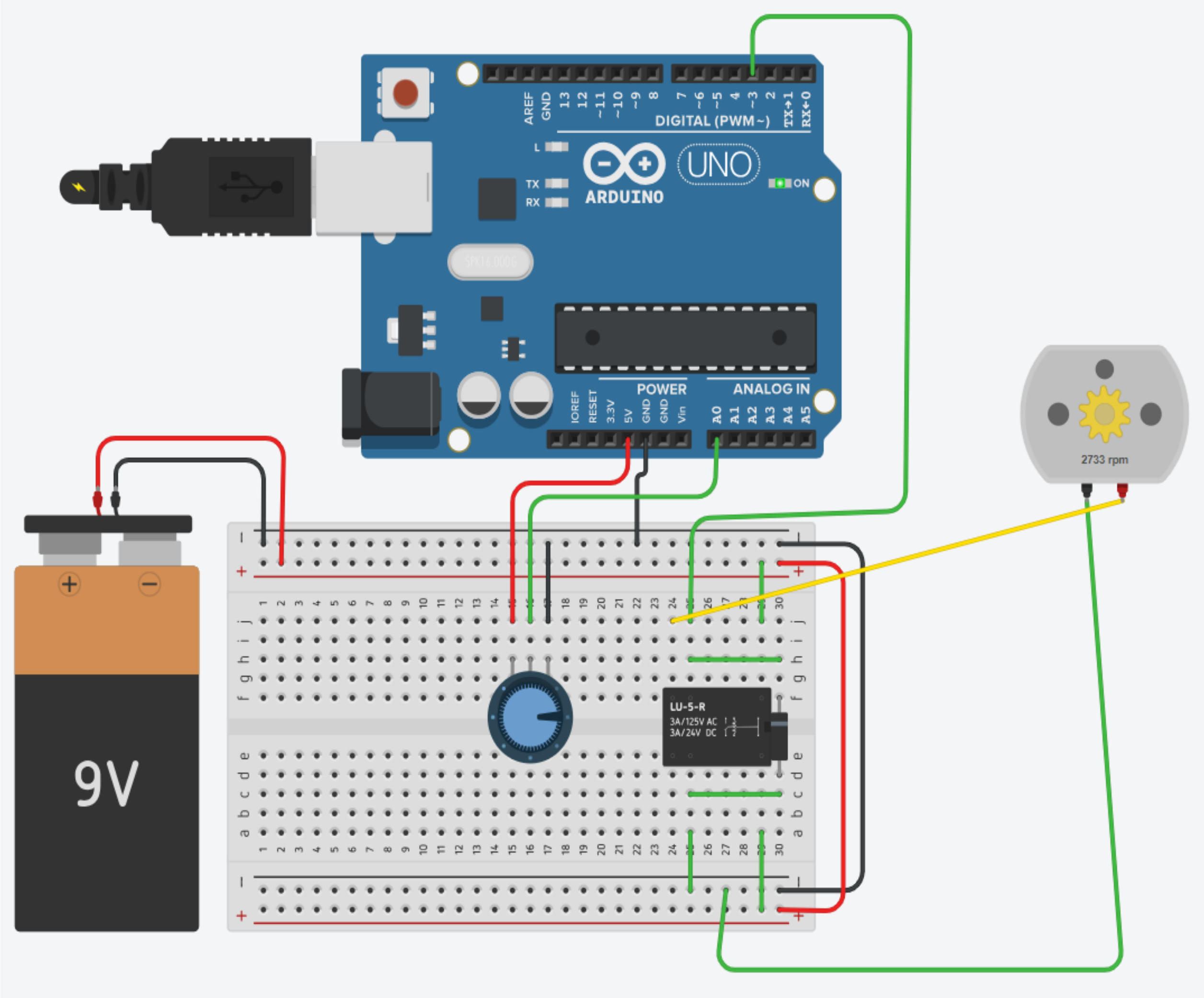
BOARD	PWM PINS *	PWM FREQUENCY
UNO (R3 and earlier), Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pins 5 and 6: 980 Hz)
UNO R4 (Minima, WiFi) *	3, 5, 6, 9, 10, 11	490 Hz
Mega	2 - 13, 44 - 46	490 Hz (pins 4 and 13: 980 Hz)
GIGA R1 **	2 - 13	500 Hz
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pins 3 and 11: 980 Hz)
UNO WiFi Rev2, Nano Every	3, 5, 6, 9, 10	976 Hz
MKR boards *	0 - 8, 10, A3, A4	732 Hz
MKR1000 WiFi **	0 - 8, 10, 11, A3, A4	732 Hz
Zero **	3 - 13, A0, A1	732 Hz
Nano 33 IoT **	2, 3, 5, 6, 9 - 12, A2, A3, A5	732 Hz
Nano 33 BLE/BLE Sense ****	1 - 13, A0 - A7	500 Hz
Due ***	2-13	1000 Hz
101	3, 5, 6, 9	pins 3 and 9: 490 Hz, pins 5 and 6: 980 Hz

Arduino: PWM Outputs (2)

- On some microcontrollers PWM is only available on selected pins. Please consider the pinout diagram of your board to find out which ones you can use for PWM. They are denoted with a tilde sign (~).
- In the graphic on the right, the green lines represent a regular period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each.
- A call to analogWrite() is on a scale of 0 - 255,
  - analogWrite(255) requests a 100% duty cycle (always on)
  - analogWrite(127) is a 50% duty cycle (on half the time)


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Arduino: Analog Read & PWM Example



```
#define POT A0
#define MOTOR 3
void setup() {
  pinMode(MOTOR,OUTPUT);
}
void loop() {
  unsigned int potValue = analogRead(POT);
  analogWrite(MOTOR, potValue >> 2);
 delay(100);
}
```

There is no actual analog output via DAC in Arduino !

LTS #1: Why we divide potValue by 4 (right-shifting 2 bits) before applying analog output?

LTS #2: If we connect the motor to the digital pin 2 instead of digital pin 3, what will be changed?

LTS #3: Instead of the line starting with analogWrite command, if we replace the following two lines, what will be changed?

```
unsigned int motorValue = map(potValue, 0, 1023, 0, 255);
analogWrite(MOTOR, motorValue);
```

Raspberry Pi Pico: Analog Read & PWM Example

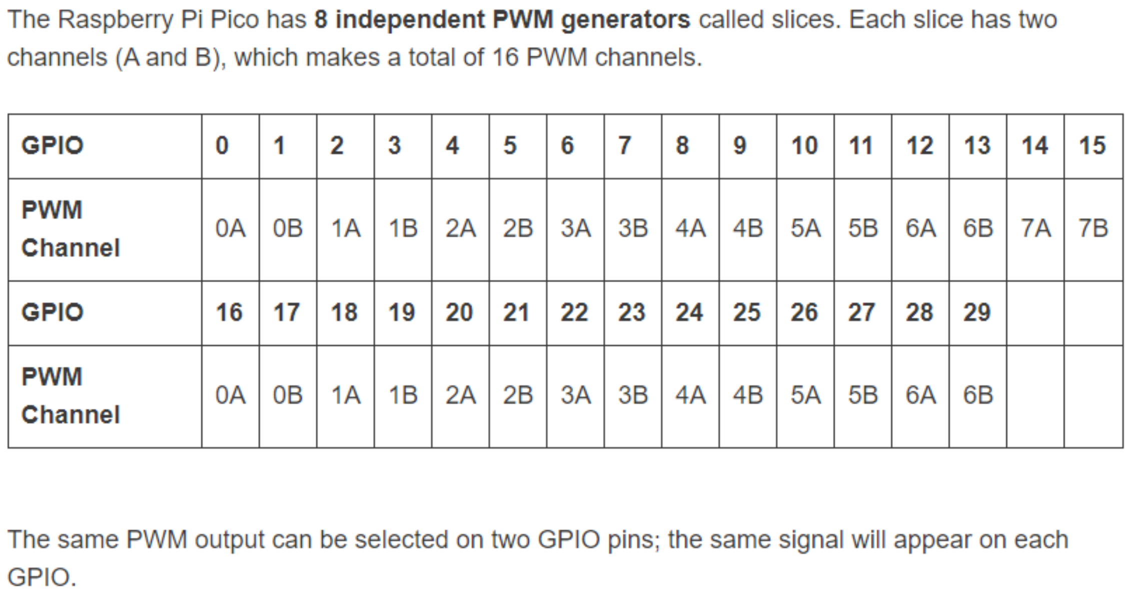
- Connecting analog input device
  - POT = AnalogIn(board.A1)
- Connecting analog output device via PWM
  - MOTOR = pwmio.PWMout(board.D5, frequency=5000, duty_cycle=0)

```
import time
import board
from analogio import AnalogIn, AnalogOut
import pwmio

POT = AnalogIn(board.A1)
MOTOR = pwmio.PWMout(board.D5, frequency=5000, duty_cycle=0)

while True:
  potValue = get_voltage(POT)
  MOTOR.duty_cycle = POT
  time.sleep(0.1)
```

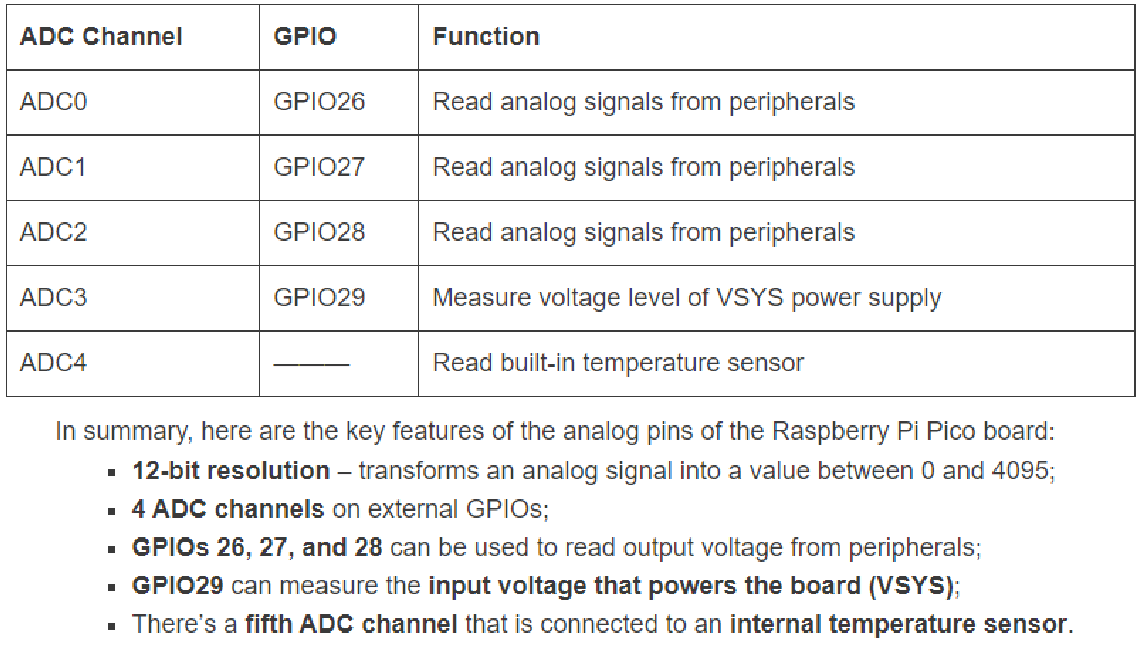
Raspberry Pi Pico: PWM & ADC Channels



The Raspberry Pi Pico has **8 independent PWM generators** called slices. Each slice has two channels (A and B), which makes a total of 16 PWM channels.

GPIO	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
PWM Channel	0A	0B	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	6A	6B	7A	7B
GPIO	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**		
PWM Channel	0A	0B	1A	1B	2A	2B	3A	3B	4A	4B	5A	5B	6A	6B		

The same PWM output can be selected on two GPIO pins; the same signal will appear on each GPIO.



ADC Channel	**GPIO**	**Function**
ADC0	GPIO26	Read analog signals from peripherals
ADC1	GPIO27	Read analog signals from peripherals
ADC2	GPIO28	Read analog signals from peripherals
ADC3	GPIO29	Measure voltage level of VSYS power supply
ADC4	———	Read built-in temperature sensor

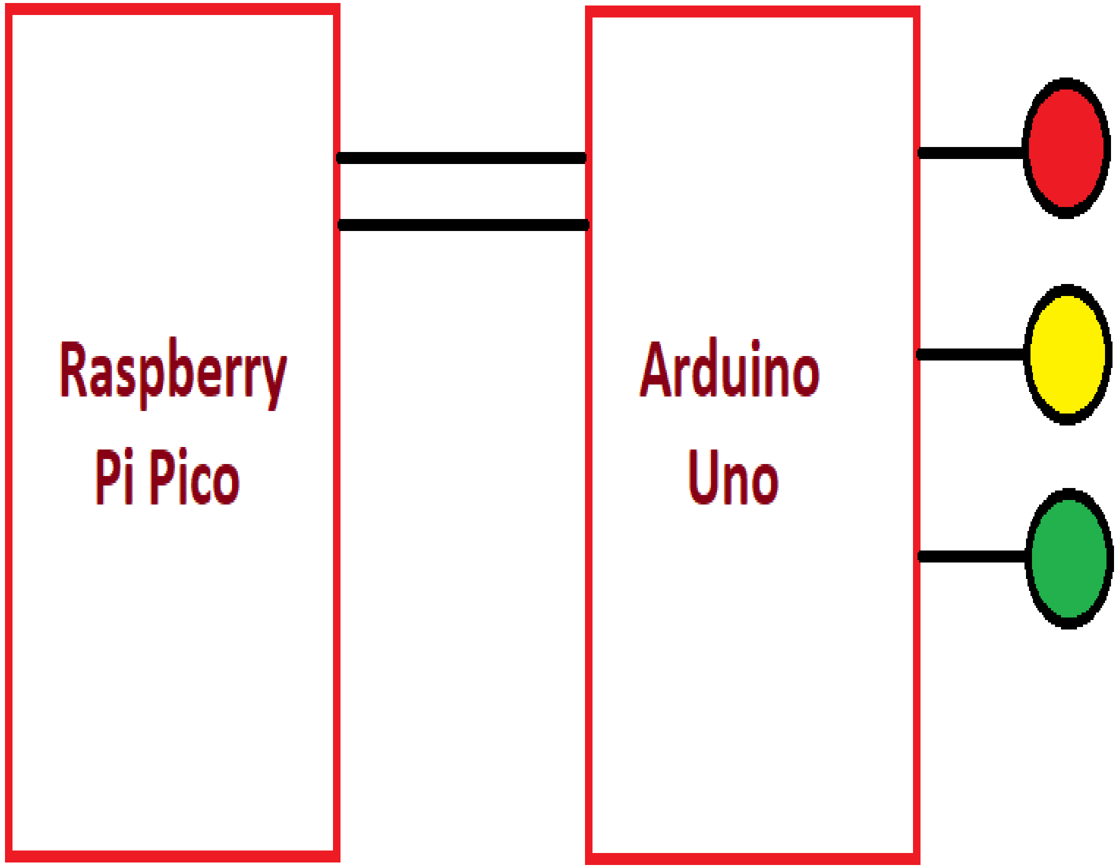
In summary, here are the key features of the analog pins of the Raspberry Pi Pico board:

- **12-bit resolution** – transforms an analog signal into a value between 0 and 4095;
- **4 ADC channels** on external GPIOs;
- **GPIOs 26, 27, and 28** can be used to read output voltage from peripherals;
- **GPIO29** can measure the **input voltage that powers the board (VSYS)**;
- There's a **fifth ADC channel** that is connected to an **internal temperature sensor**.

There is no actual analog output via DAC in Raspberry Pi Pico, too !

LTS: Raspberry Pi Pico & Arduino Uno Comnnection

- Establish a serial communication between Raspberry Pi Pico (CircuitPython) and Arduino Uno (Arduino IDE C).
- In every 10 seconds, Arduino sends the Rpi Pico «T?» via serial port.
- Whenever Rpi Pico reads «T?» from the serial port,
  - it reads its internal temperature sensor value,
  - and sends the temperature in the format of «T=25.6» to the Arduino via serial port (where 25.6 is the example temperature value)
- Whenever Arduino reads the temperature value from the serial port,
  - if this value is less than 30.0, it lightens green LED
  - if this value is greater than or equals to 30.0 but less 45.0, it lightens yellow LED
  - otherwise, it lightens red LED.



Thanks for listening ☺

YALÇIN İŞLER

Assoc. Prof.