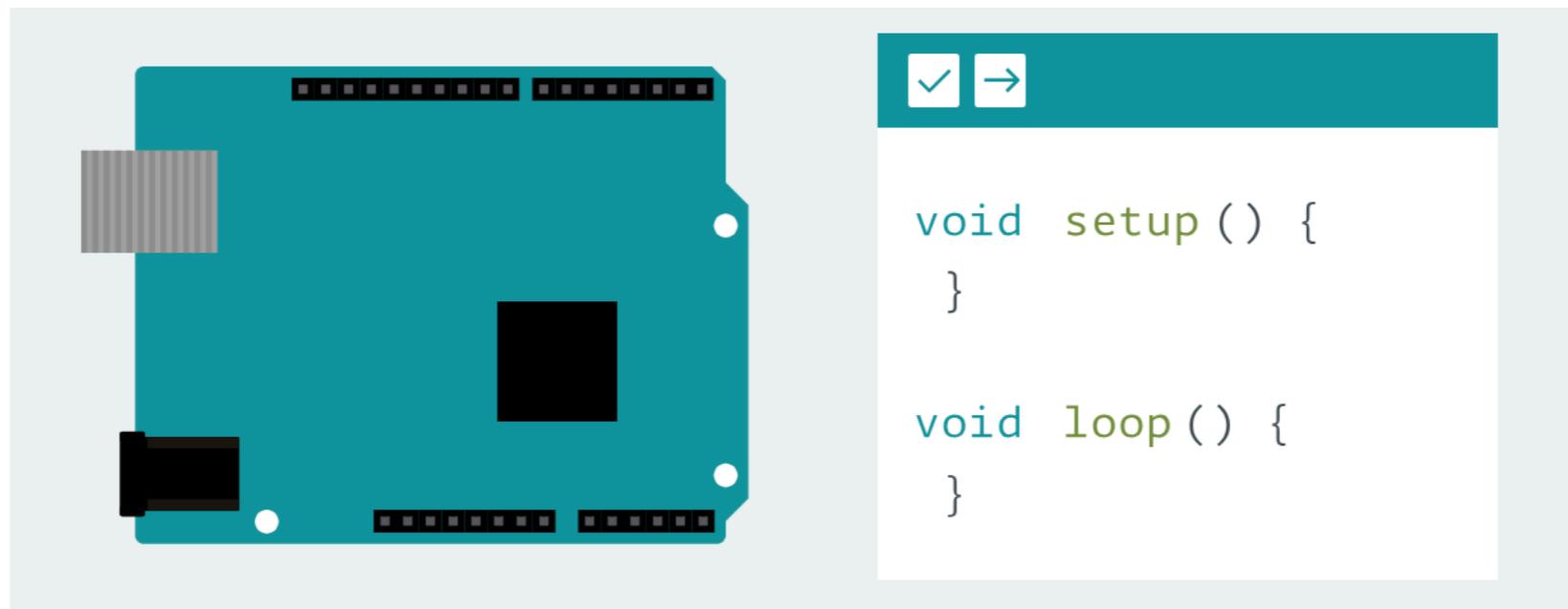


Lecture #5

Arduino

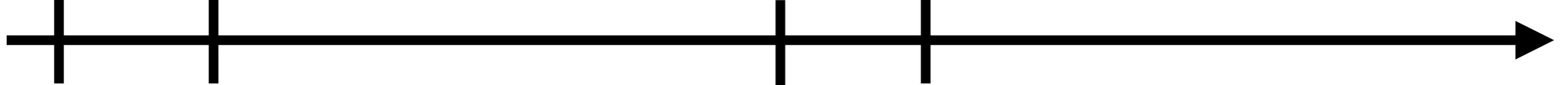
Android Things 2019

Foundations



History





1972

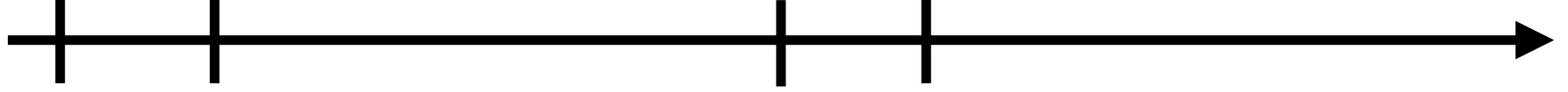
1982

1999

2003

History





1972 1982

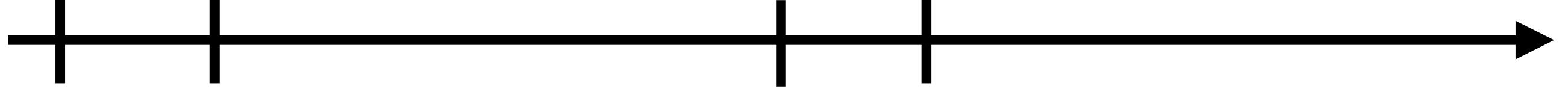
1999

2003

History



Arduino d'Ivrea
(Civica raccolta stampe - Milano)



1972 1982

1999

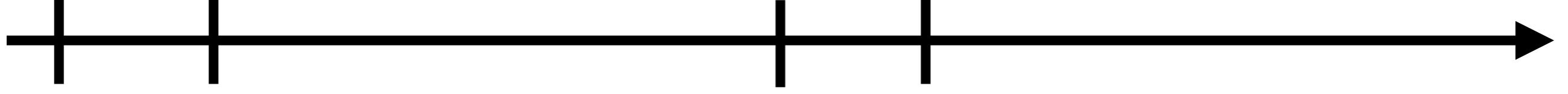
2003

History



Arduino d'Ivrea
(Civica raccolta stampe - Milano)

King of Italy (1002–1014)



1972 1982

1999

2003

History



Arduino d'Ivrea
(Civica raccolta stampe - Milano)

[King of Italy](#) (1002–1014)

https://en.wikipedia.org/wiki/Arduin_of_Ivrea

History



Hernando Barragán

History

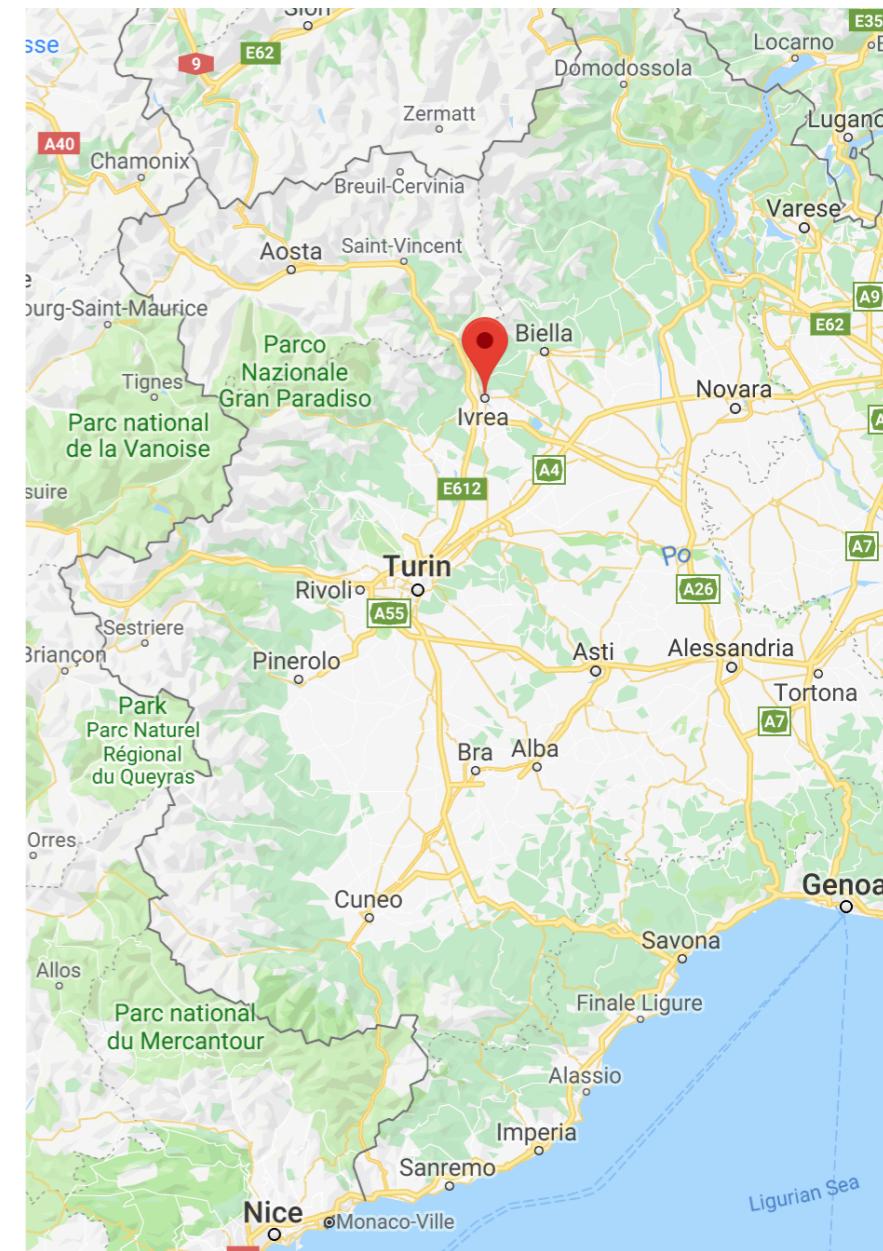


Hernando Barragán

History



Hernando Barragán



Interaction Design Institute Ivrea
IDII

History



History



Massimo Banzi

History



Casey Reas



Massimo Banzi

History



Casey Reas



2001

<https://processing.org/>



Massimo Banzi

History



```
void setup() {
    size(400, 400);
    stroke(255);
    background(192, 64, 0);
}

void draw() {
    line(150, 25, mouseX, mouseY);
}
```

History



```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}
```

History



Wiring

2003

```
void setup() {  
  size(400, 400);  
  stroke(255);  
  background(192, 64, 0);  
}  
  
void draw() {  
  line(150, 25, mouseX, mouseY);  
}
```

History



Wiring

2003

```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}  
  
int ledPin = WLED; // a name for the on-board LED  
  
void setup () {  
    pinMode(ledPin, OUTPUT); // configure the pin for digital output  
}  
  
void loop () {  
    digitalWrite(ledPin, HIGH); // turn on the LED  
    delay (1000);  
    digitalWrite(ledPin, LOW); // turn off the LED  
    delay (1000); // wait one second (1000 milliseconds)  
}
```

History



Wiring

2003

```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}  
  
int ledPin = WLED; // a name for the on-board LED  
  
void setup () {  
    pinMode(ledPin, OUTPUT); // configure the pin for digital output  
}  
  
void loop () {  
    digitalWrite(ledPin, HIGH); // turn on the LED  
    delay (1000); // wait one second (1000 milliseconds)  
    digitalWrite(ledPin, LOW); // turn off the LED  
    delay (1000); // wait one second  
}
```

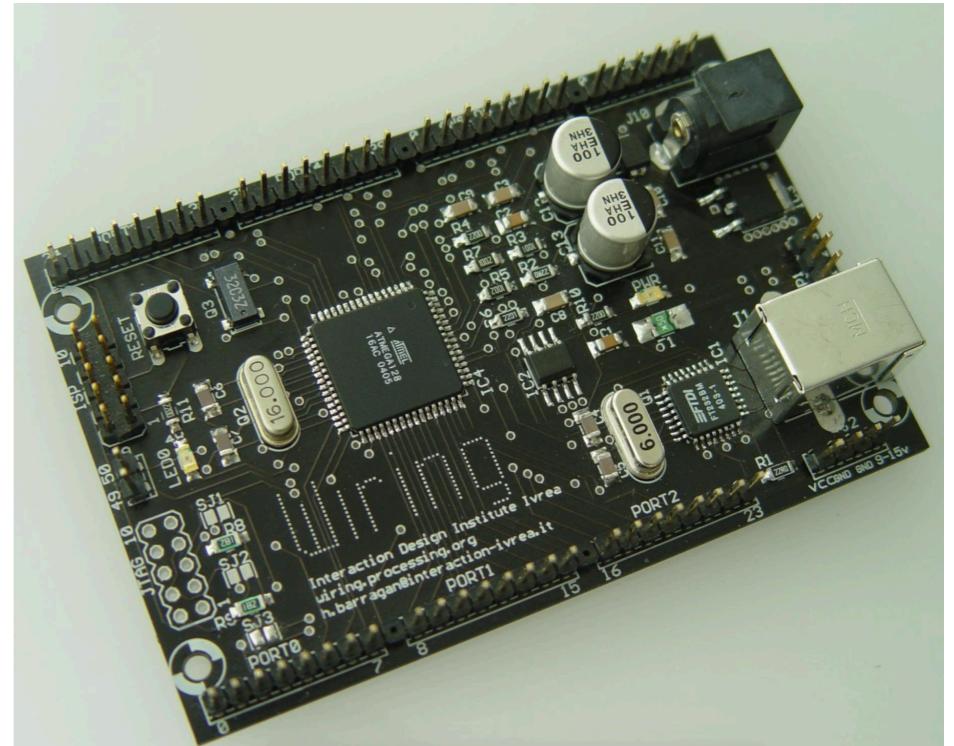
History



Wiring

2003

```
void setup() {  
    size(400, 400);  
    stroke(255);  
    background(192, 64, 0);  
}  
  
void draw() {  
    line(150, 25, mouseX, mouseY);  
}  
  
int ledPin = WLED; // a name for the on-board LED  
  
void setup () {  
    pinMode(ledPin, OUTPUT); // configure the pin for digital output  
}  
  
void loop () {  
    digitalWrite(ledPin, HIGH); // turn on the LED  
    delay (1000); // wait one second (1000 milliseconds)  
    digitalWrite(ledPin, LOW); // turn off the LED  
    delay (1000); // wait one second  
}
```

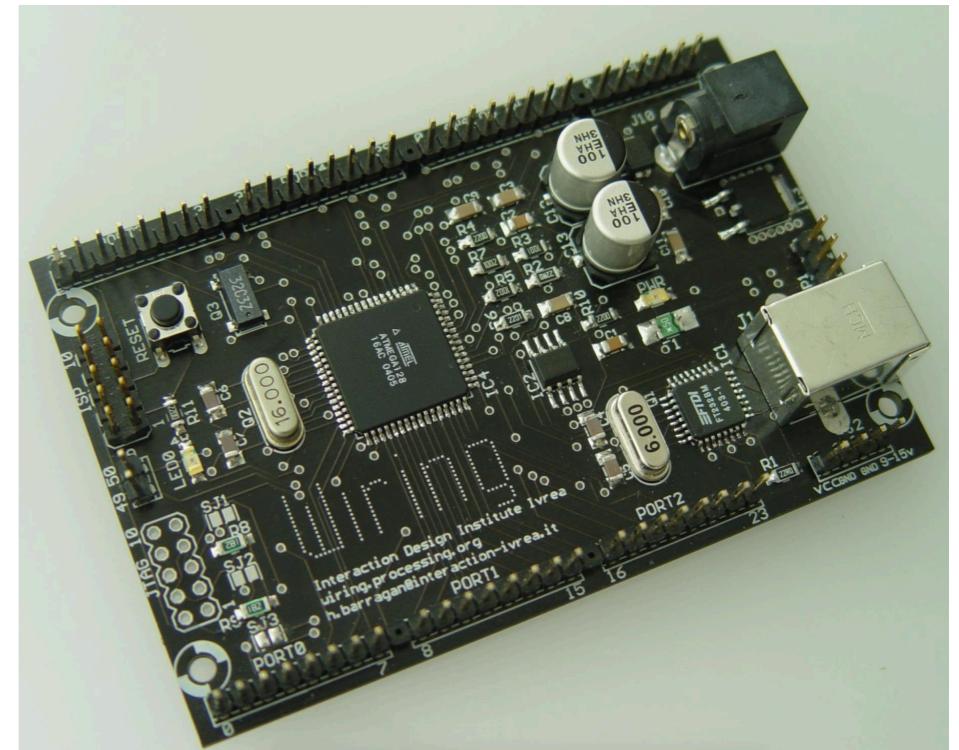


History



Wiring

2003



```
int ledPin = WLED; // a name for the on-board LED

void setup () {
    pinMode(ledPin, OUTPUT); // configure the pin for digital output
}

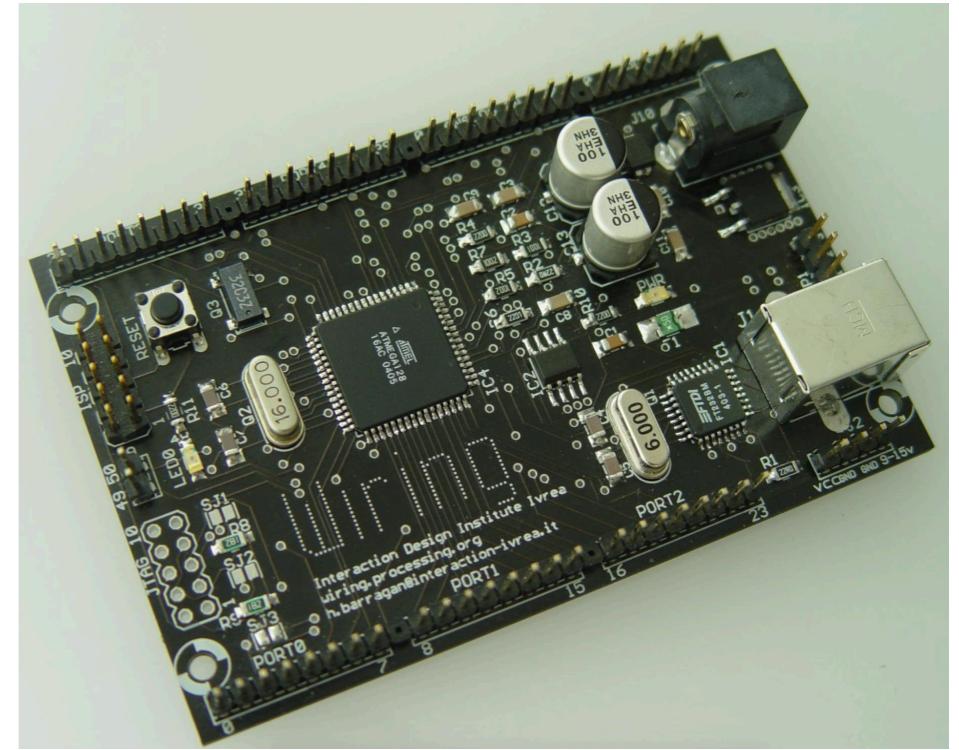
void loop () {
    digitalWrite(ledPin, HIGH); // turn on the LED
    delay (1000); // wait one second (1000 milliseconds)
    digitalWrite(ledPin, LOW); // turn off the LED
    delay (1000); // wait one second
}
```

History



Wiring

2003



```
int ledPin = WLED; // a name for the on-board LED

void setup () {
    pinMode(ledPin, OUTPUT); // configure the pin for digital output}

void loop () {
    digitalWrite(ledPin, HIGH); // turn on the LED
    delay (1000);
    digitalWrite(ledPin, LOW); // turn off the LED
    delay (1000);
}
```

History



Wiring

2003

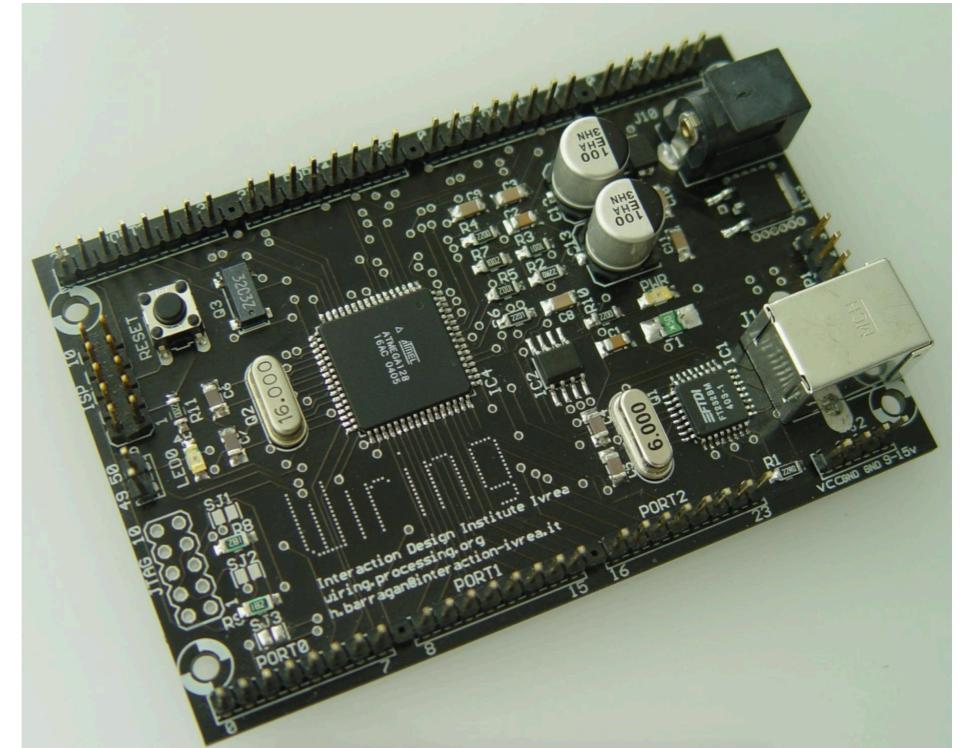


```
int ledPin = WLED; // a name for the on-board LED

void setup () {
  pinMode(ledPin, OUTPUT); // configure the pin for digital output

}

void loop () {
  digitalWrite(ledPin, HIGH); // turn on the LED
  delay (1000);
  digitalWrite(ledPin, LOW); // turn off the LED
  delay (1000); // wait one second (1000 milliseconds)
}
```

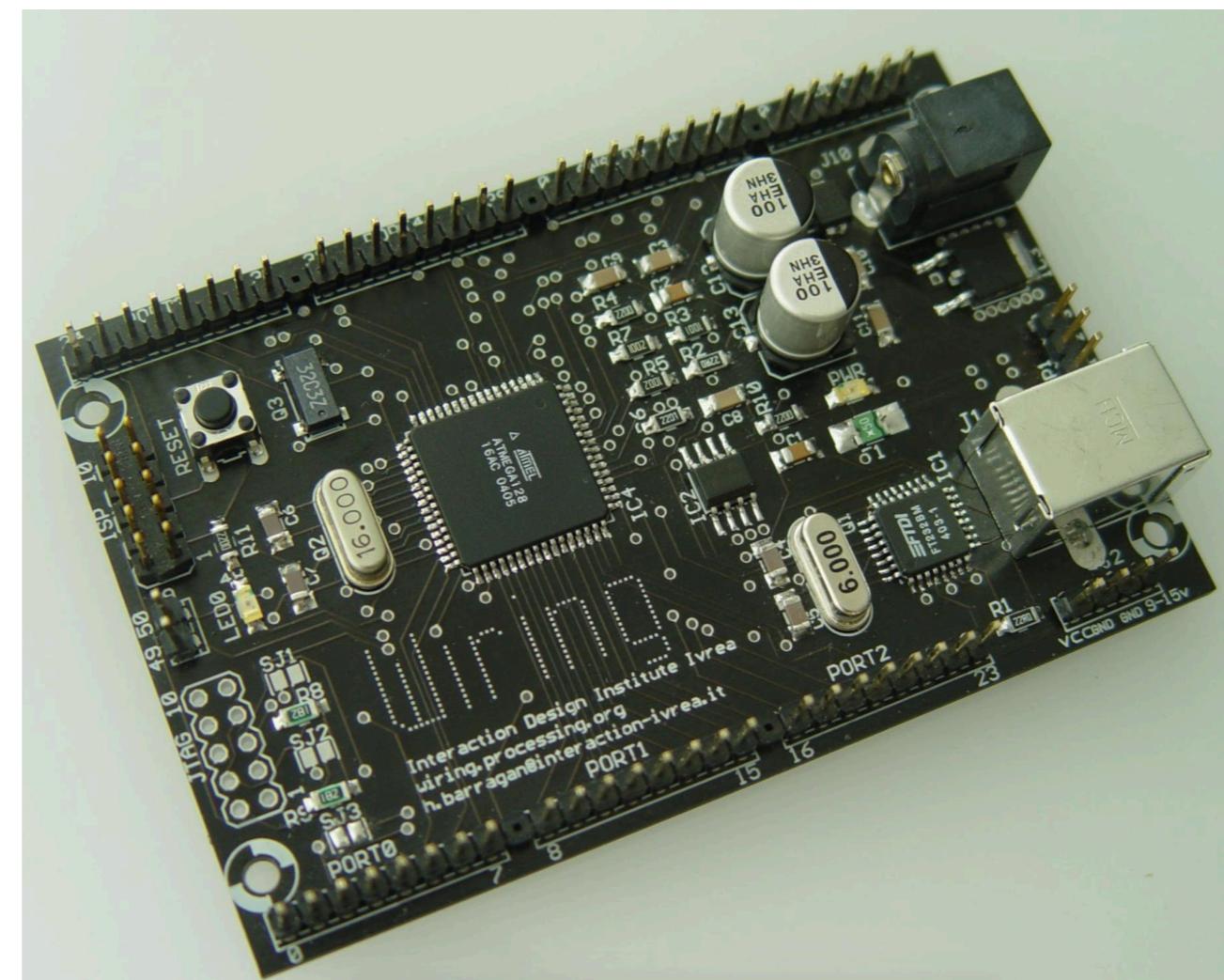


History



Wiring

2003

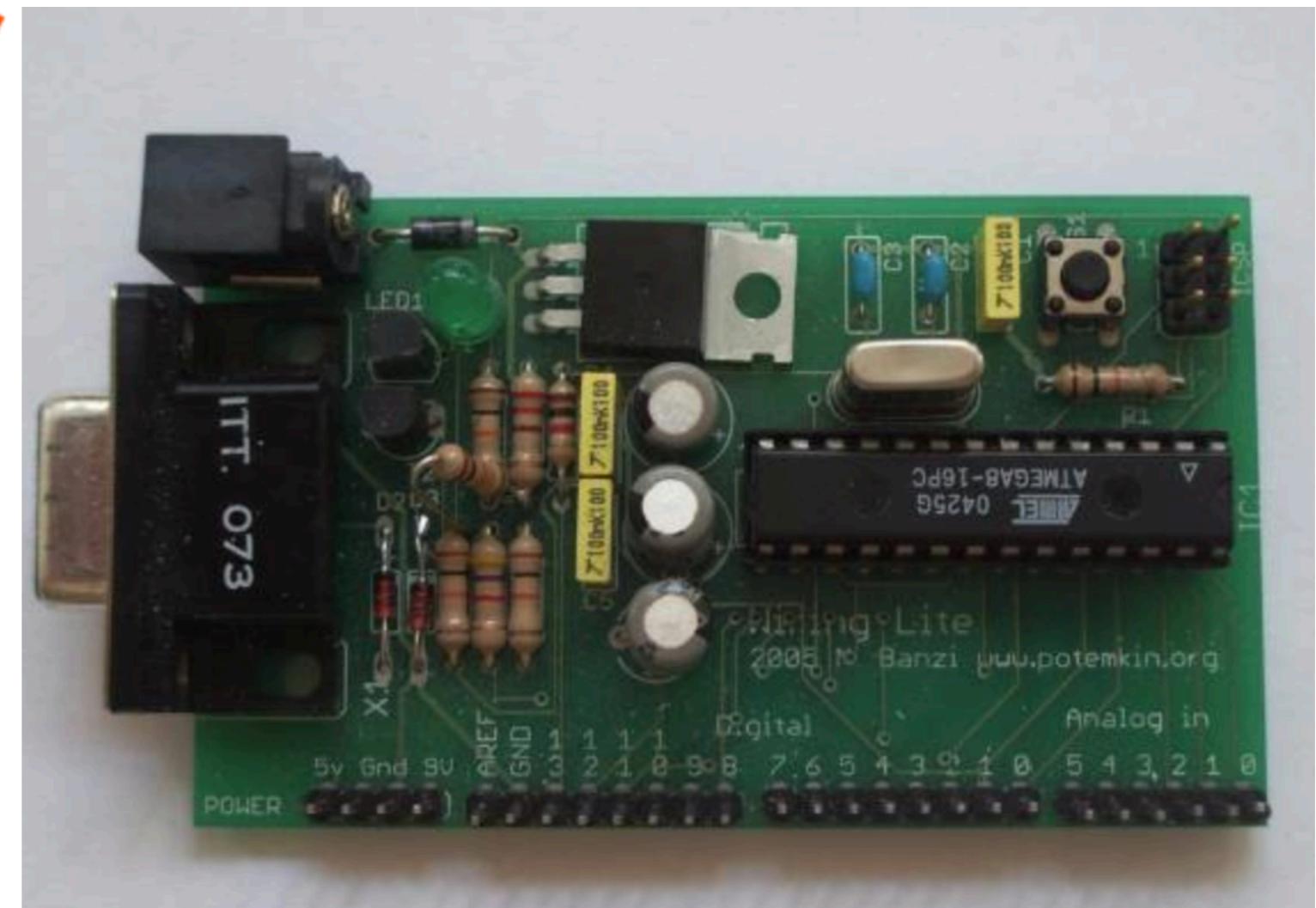


History



Wiring

2003



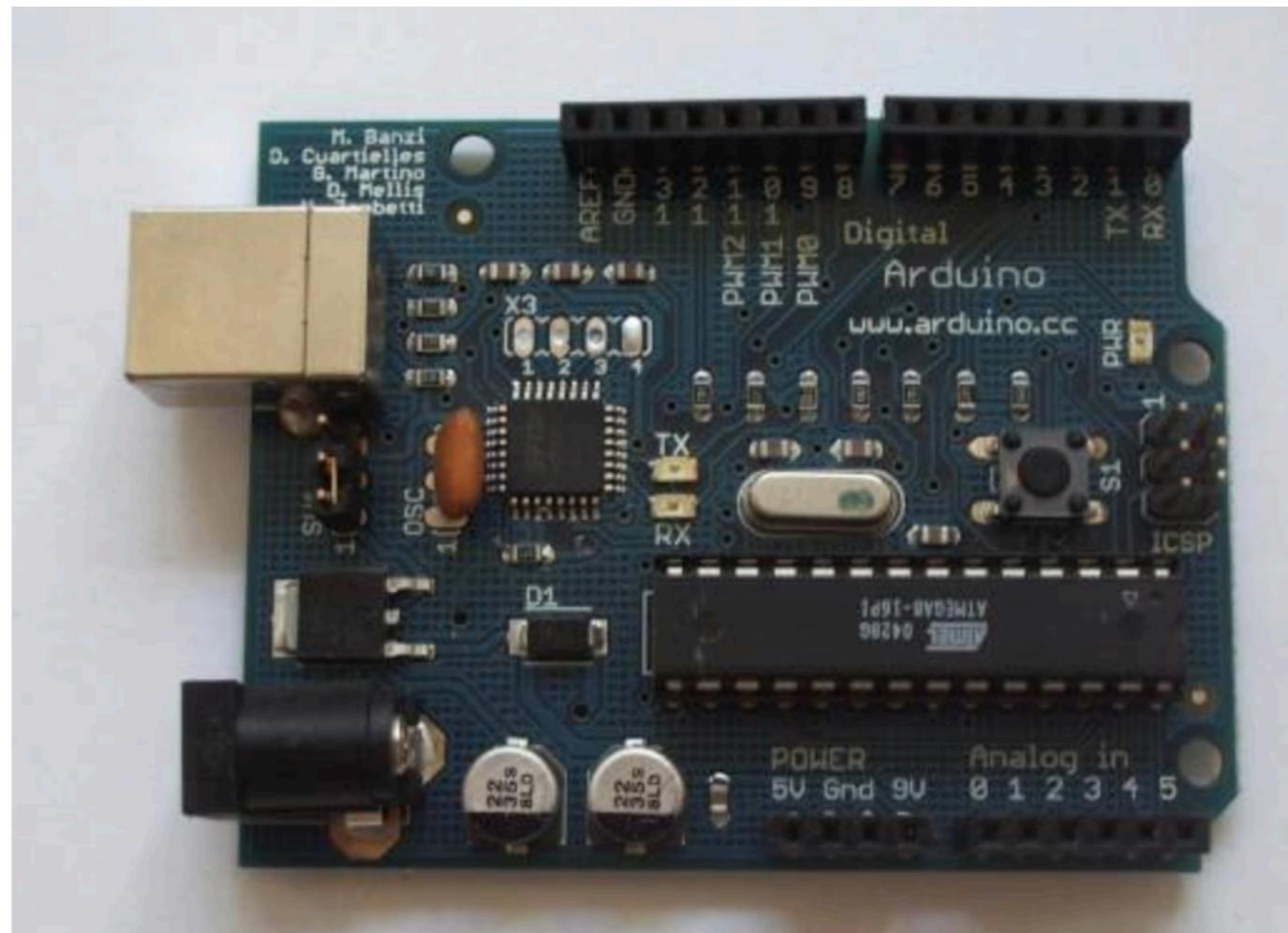
2004

History



Wiring

2003



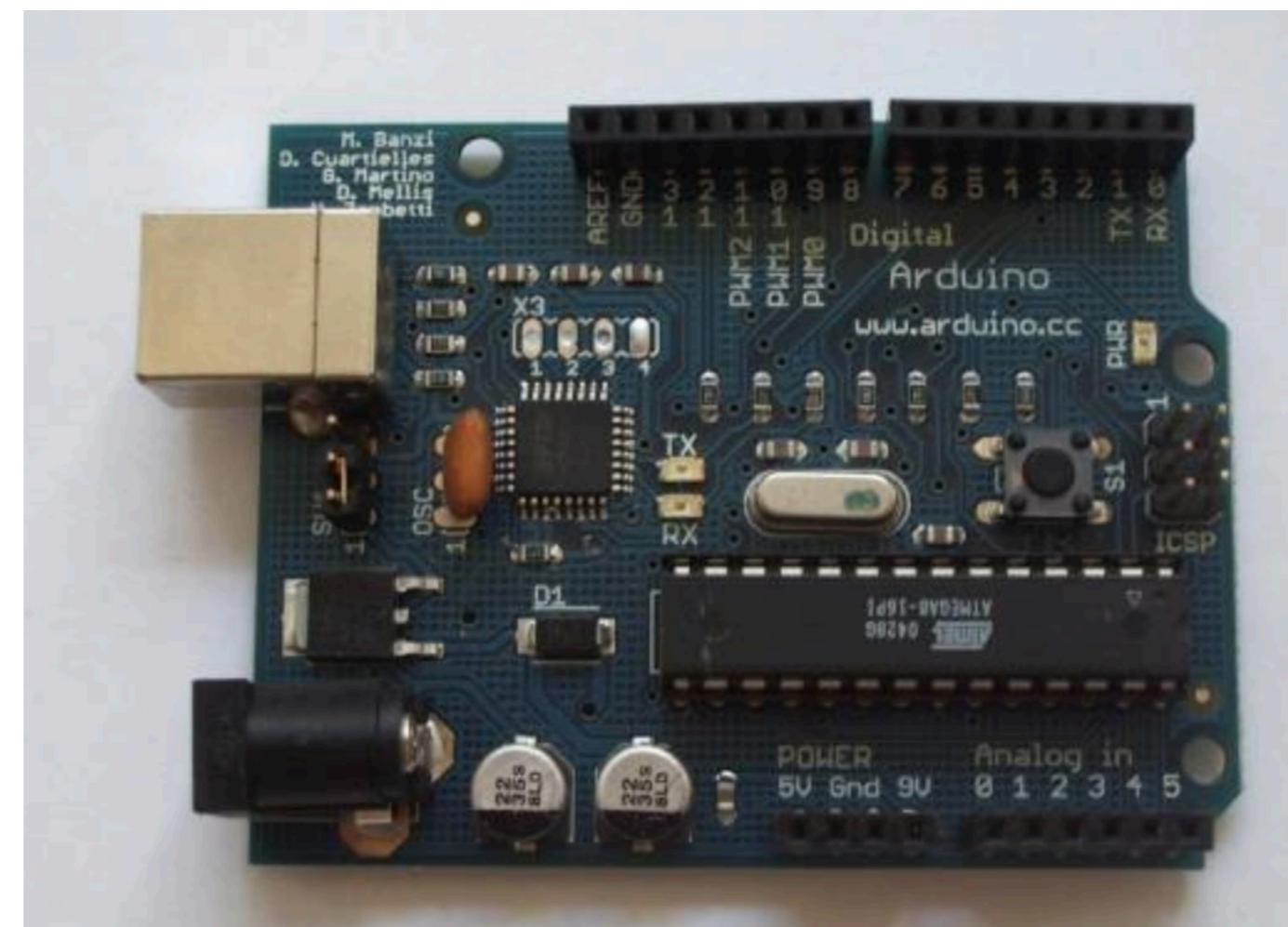
2004

History



Wiring

2003



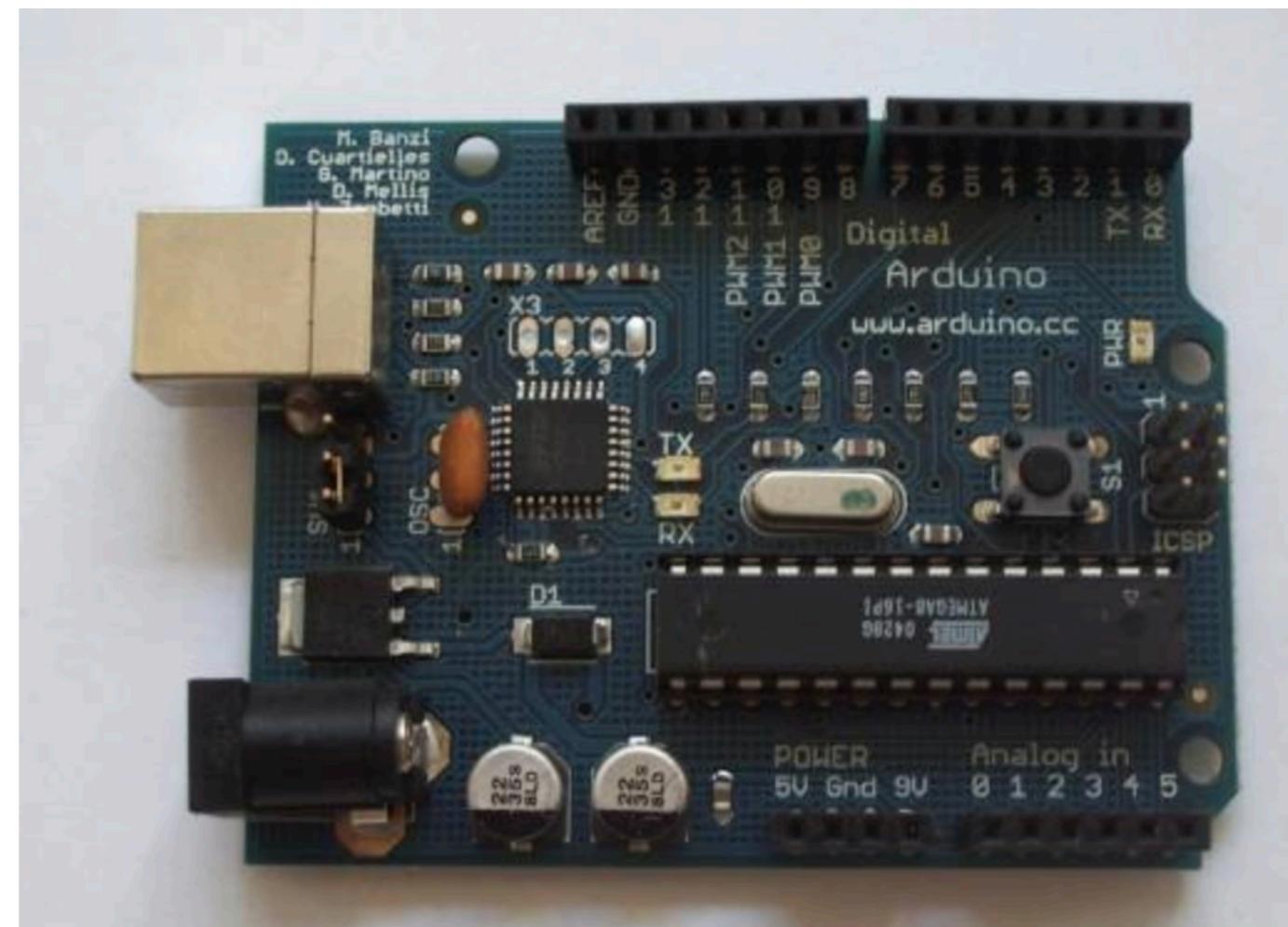
2004

History



Wiring

2003



2004

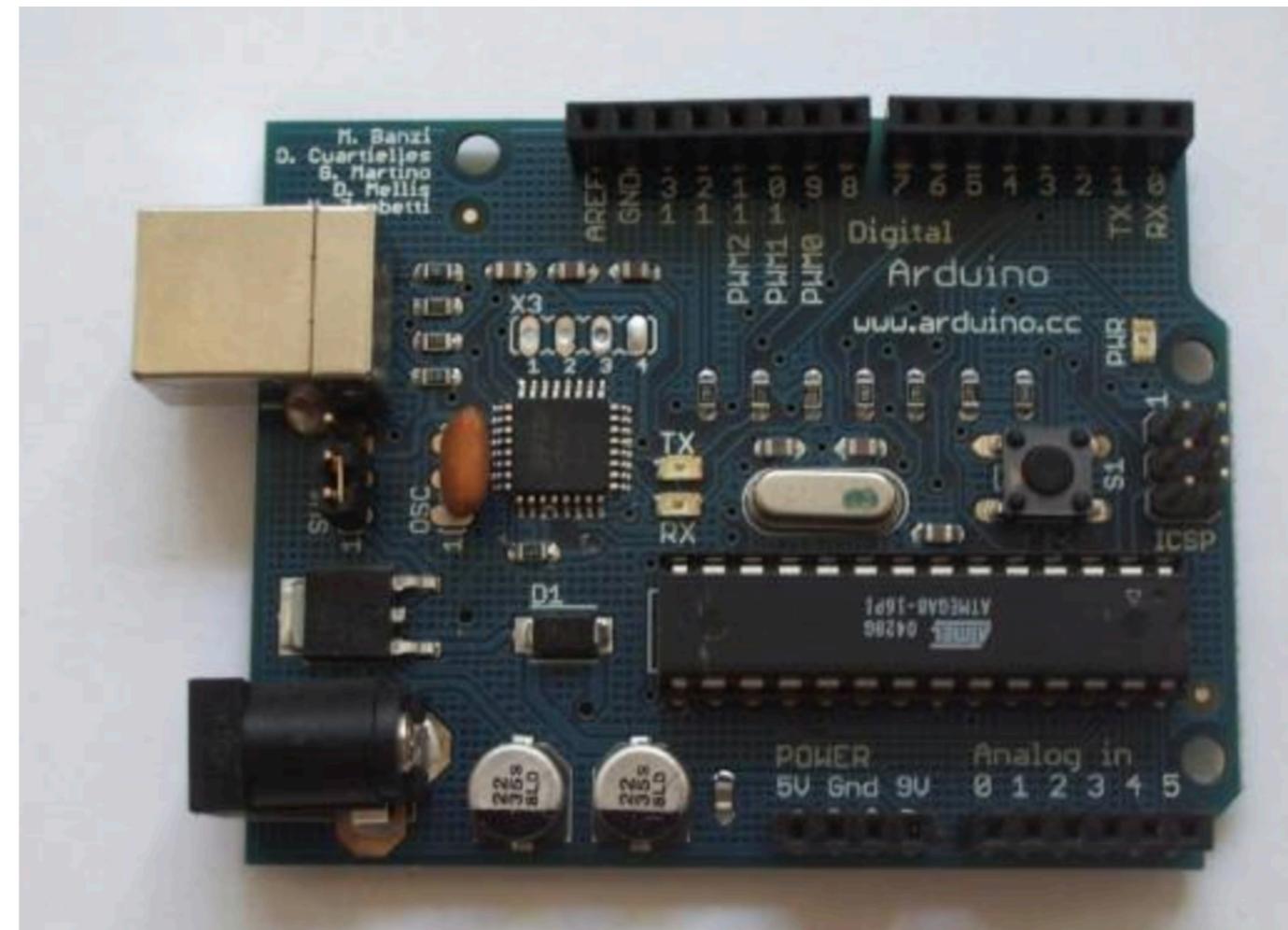
Arduino

History



Wiring

2003



2004

Arduino

<https://arduinohistory.github.io/>

Credits

Originally started as a research project by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis at the Interaction Design Institute of Ivrea in the early 2000s, it builds upon the Processing project, a language for learning how to code within the context of the visual arts developed by Casey Reas and Ben Fry as well as a thesis project by Hernando Barragan about the Wiring board.

Credits

Originally started as a research project by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis at the Interaction Design Institute of Ivrea in the early 2000s, it builds upon the Processing project, a language for learning how to code within the context of the visual arts developed by Casey Reas and Ben Fry as well as a thesis project by Hernando Barragan about the Wiring board.

Language

Language

- Functions

Digital I/O

`digitalRead()`

`digitalWrite()`

`pinMode()`

Language

- Functions

Digital I/O

`digitalRead()`

`digitalWrite()`

`pinMode()`

Analog I/O

`analogRead()`

`analogReference()`

`analogWrite()`

Language

- Functions

Digital I/O

`digitalRead()`

`digitalWrite()`

`pinMode()`

Analog I/O

`analogRead()`

`analogReference()`

`analogWrite()`

Time

`delay()`

`delayMicroseconds()`

`micros()`

`millis()`

Language

- Functions

Digital I/O	Time	Characters	Bits and Bytes
<code>digitalRead()</code>	<code>delay()</code>	<code>isAlpha()</code>	<code>bit()</code>
<code>digitalWrite()</code>	<code>delayMicroseconds()</code>	<code>isAlphaNumeric()</code>	<code>bitClear()</code>
<code>pinMode()</code>	<code>micros()</code>	<code>isAscii()</code>	<code>bitRead()</code>
Analog I/O	<code>millis()</code>	<code>isControl()</code>	<code>bitSet()</code>
	Math	<code>isDigit()</code>	<code>bitWrite()</code>
<code>analogRead()</code>	<code>abs()</code>	<code>isGraph()</code>	<code>highByte()</code>
<code>analogReference()</code>	<code>constrain()</code>	<code>isHexadecimalDigit()</code>	<code>lowByte()</code>
<code>analogWrite()</code>	<code>map()</code>	<code>isLowerCase()</code>	External Interrupts
Zero, Due & MKR Family	<code>max()</code>	<code>isPrintable()</code>	<code>attachInterrupt()</code>
<code>analogReadResolution()</code>	<code>min()</code>	<code>isPunct()</code>	<code>detachInterrupt()</code>
<code>analogWriteResolution()</code>	<code>pow()</code>	<code>isSpace()</code>	Interrupts
Advanced I/O	<code>sq()</code>	<code>isUpperCase()</code>	<code>interrupts()</code>
<code>noTone()</code>	<code>sqrt()</code>	<code>isWhitespace()</code>	<code>noInterrupts()</code>
<code>pulseIn()</code>	Trigonometry	Random Numbers	Communication
<code>pulseInLong()</code>	<code>cos()</code>	<code>random()</code>	Serial
<code>shiftIn()</code>	<code>sin()</code>	<code>randomSeed()</code>	Stream
<code>shiftOut()</code>	<code>tan()</code>		USB
<code>tone()</code>			Keyboard
			Mouse

Language

- Functions
- Variables

Constants	Data Types	Variable Scope & Qualifiers
Floating Point Constants	String()	const
Integer Constants	array	scope
HIGH LOW	bool	static
INPUT OUTPUT INPUT_PULLUP	boolean	volatile
LED_BUILTIN	byte	
true false	char	Utilities
Conversion	double	PROGMEM
byte()	float	sizeof()
char()	int	
float()	long	
int()	short	
long()	size_t	
word()	string	
	unsigned char	
	unsigned int	
	unsigned long	
	void	
	word	

Language

- Functions
 - Sketch
 - loop()
 - setup()
- Variables
- Structure

Language

- Functions Sketch
 loop()
 setup()
- Variables Control Structure
- Structure break
 continue
 do...while
 else
 for
 goto
 if
 return
 switch...case
 while

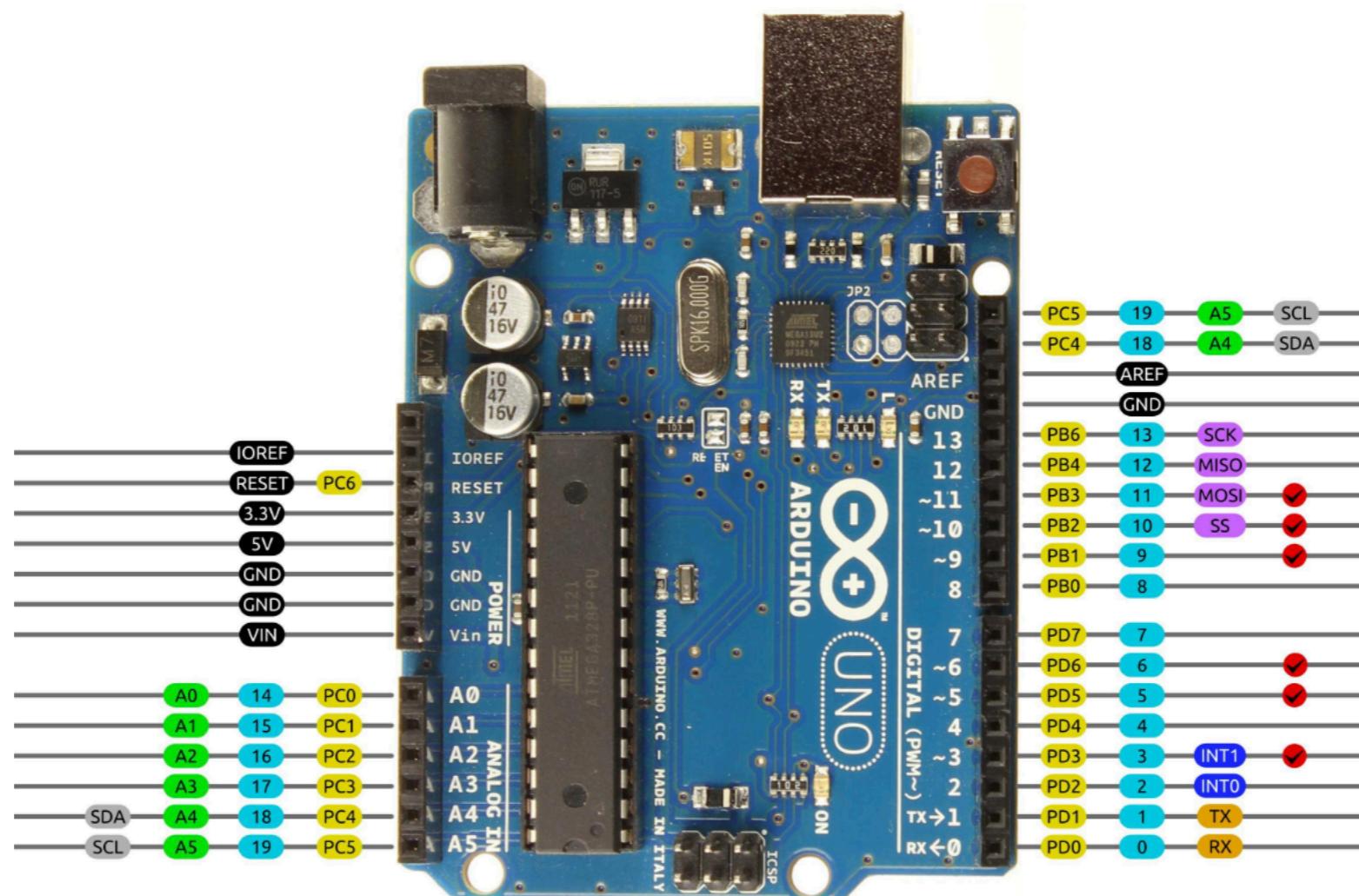
Language

- Functions
- Variables
- Structure

Sketch	Arithmetic Operators	Pointer Access Operators
loop()	% (remainder)	& (reference operator)
setup()	* (multiplication)	* (dereference operator)
Control Structure	+ (addition)	Bitwise Operators
break	- (subtraction)	& (bitwise and)
continue	/ (division)	<< (bitshift left)
do...while	= (assignment operator)	>> (bitshift right)
else	Comparison Operators	^ (bitwise xor)
for	!= (not equal to)	(bitwise or)
goto	< (less than)	~ (bitwise not)
if	<= (less than or equal to)	Compound Operators
return	== (equal to)	%= (compound remainder)
switch...case	> (greater than)	&= (compound bitwise and)
while	>= (greater than or equal to)	*= (compound multiplication)
Further Syntax	Boolean Operators	+= (increment)
#define (define)	! (logical not)	+= (compound addition)
#include (include)	&& (logical and)	-- (decrement)
/* */ (block comment)	(logical or)	-= (compound subtraction)
// (single line comment)		/= (compound division)
;		^= (compound bitwise xor)
{}		= (compound bitwise or)

Digital Pins

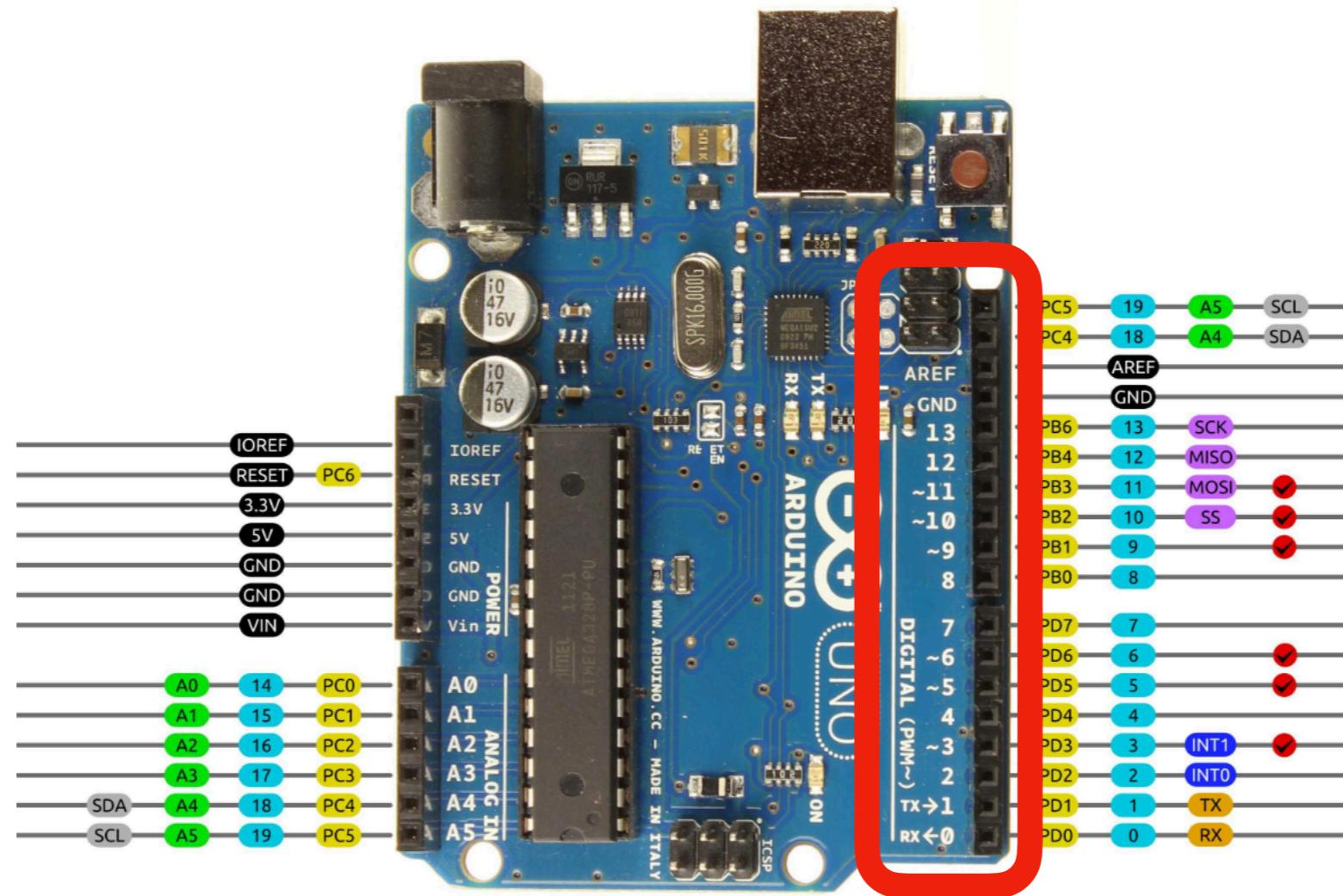
Arduino Uno R3 Pinout



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT

Digital Pins

Arduino Uno R3 Pinout



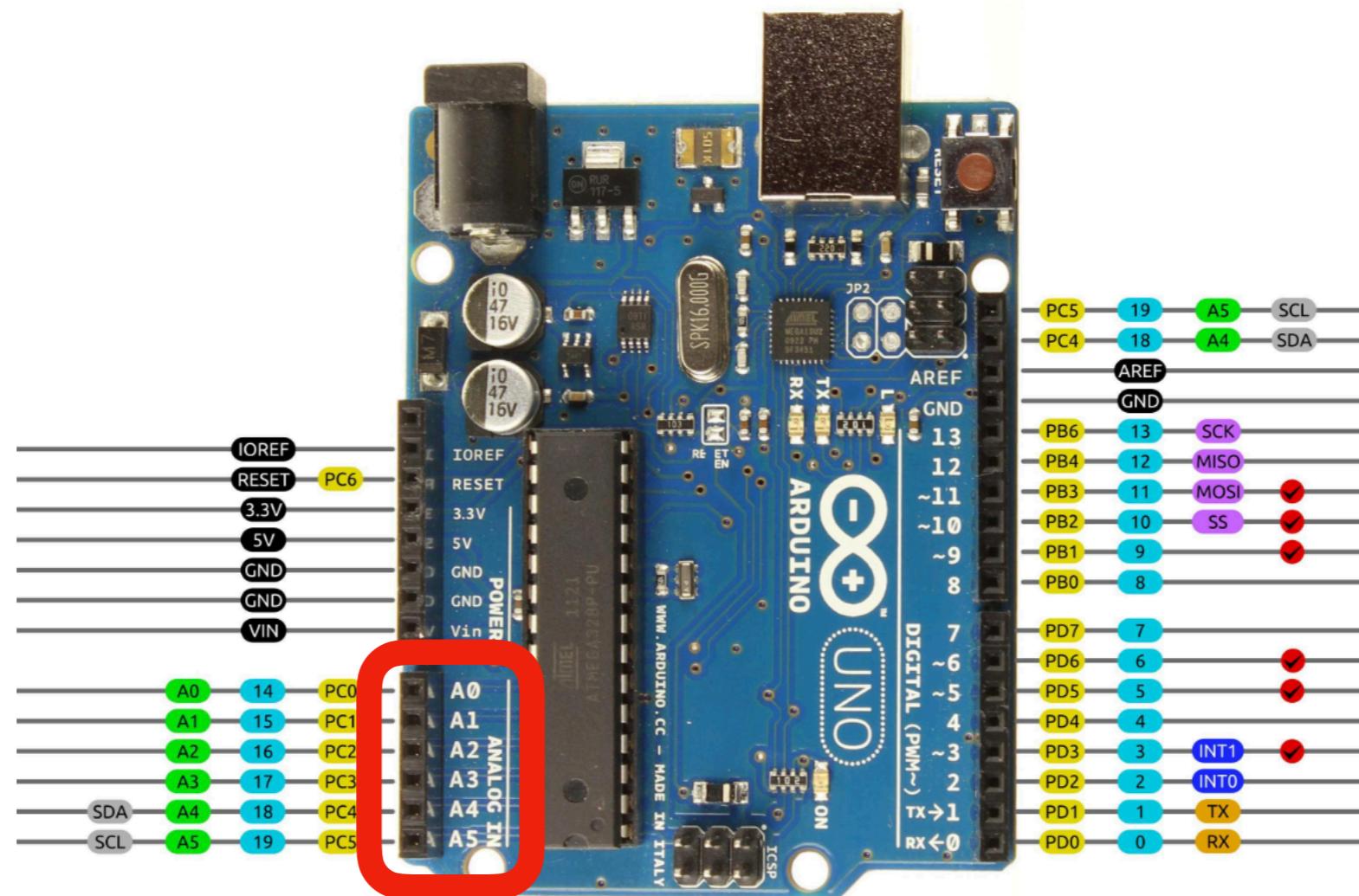
AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT



2014 by Bouni
Photo by Arduino.cc

Analog Pins

Arduino Uno R3 Pinout



AVR DIGITAL ANALOG POWER SERIAL SPI I2C PWM INTERRUPT



2014 by Bouni
Photo by Arduino.cc

Accessing the pins

```
void setup() {
    pinMode(13, OUTPUT); // sets the digital pin 13 as output
}

void loop() {
    digitalWrite(13, HIGH); // sets the digital pin 13 on
    delay(1000);          // waits for a second
    digitalWrite(13, LOW); // sets the digital pin 13 off
    delay(1000);          // waits for a second
}
```

Accessing the pins

```
int ledPin = 13; // LED connected to digital pin 13
int inPin = 7; // pushbutton connected to digital pin 7
int val = 0; // variable to store the read value

void setup() {
    pinMode(ledPin, OUTPUT); // sets the digital pin 13 as output
    pinMode(inPin, INPUT); // sets the digital pin 7 as input
}

void loop() {
    val = digitalRead(inPin); // read the input pin
    digitalWrite(ledPin, val); // sets the LED to the button's value
}
```

Accessing the pins

```
int analogPin = A3; // potentiometer wiper (middle terminal) connected to analog pin 3
                    // outside leads to ground and +5V
int val = 0; // variable to store the value read

void setup() {
    Serial.begin(9600); // setup serial
}

void loop() {
    val = analogRead(analogPin); // read the input pin
    Serial.println(val); // debug value
}
```

Accessing the pins

```
int ledPin = 9;      // LED connected to digital pin 9
int analogPin = 3;   // potentiometer connected to analog pin 3
int val = 0;         // variable to store the read value

void setup() {
  pinMode(ledPin, OUTPUT); // sets the pin as output
}

void loop() {
  val = analogRead(analogPin); // read the input pin
  analogWrite(ledPin, val / 4);
  // analogRead values go from 0 to 1023, analogWrite values from 0 to 255
}
```

Connect to Android Things

```
private val uart: UartDevice by lazy {
    PeripheralManager.getInstance().openUartDevice(uartDevice).apply {
        setBaudrate(115200)
        setDataSize(8)
        setParity(UartDevice.PARITY_NONE)
        setStopBits(1)
    }
}
```

Connect to Android Things

```
fun read(): String {
    val maxCount = 8
    val buffer = ByteArray(maxCount)
    var output = ""
    do {
        val count = uart.read(buffer, buffer.size)
        output += buffer.toReadableString()
        if (count == 0) break
        logd("Read ${buffer.toReadableString()} $count bytes from
peripheral")
    } while (true)
    return output
}

private fun ByteArray.toReadableString() = filter { it > 0.toByte() }
    .joinToString(separator = "") { it.toChar().toString() }
```

```
fun read(): String {
    val maxCount = 8
    val buffer = ByteArray(maxCount)
    var output = ""
    do {
        val count = uart.read(buffer, buffer.size)
        output += buffer.toReadableString()
        if (count == 0) break
        logd("Read ${buffer.toReadableString()} $count bytes from
peripheral")
    } while (true)
    return output
}

private fun ByteArray.toReadableString() = filter { it > 0.toByte() }
    .joinToString(separator = "") { it.toChar().toString() }

fun write(value: String) {
    val count = uart.write(value.toByteArray(), value.length)
    logd("Wrote $value $count bytes to peripheral")
}
```

Connect to Android Things

```
void setup() {  
    Serial.begin(115200);  
}
```

```
void loop() {  
    processSerialCommand();  
}
```

Connect to Android Things

```
void processSerialCommand() {  
    if (Serial.available() > 0) {  
        char command = (char) Serial.read();  
        switch (command) {  
            case 'X':  
                doStuff();  
                break;  
        }  
        Serial.flush();  
    }  
}  
  
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    processSerialCommand();  
}
```

Connect to Android Things

```
void processSerialCommand() {  
    if (Serial.available() > 0) {  
        char command = (char) Serial.read();  
        switch (command) {  
            case 'X':  
                doStuff();  
                break;  
        }  
        Serial.flush();  
    }  
}  
  
void setup() {  
    Serial.begin(115200);  
}  
  
void loop() {  
    processSerialCommand();  
}
```

```
void doStuff() {  
    int sensorValue = analogRead(A0);  
    sprintf(buff, "%d", sensorValue);  
    Serial.write(buff);  
}
```

Lecture outcomes

- Using Arduino platform.
- Link Android Things boards with Arduino.
- Capture sensors data.

