Welcome to

Embedded Systems

Prof. Dr. Volker Strumpen

Rhine-Waal University of Applied Sciences

Winter Semester 2019

Organization

Moodle: CI_5.01 self enrollment key is `embedDIY`

Literature available for download on Moodle

A. Arduino

1. Fitzgerald and Shiloh, *The Arduino Projects Book*, 2012 [slightly outdated], see `arduino.cc/starterkit`.
2. Pan and Zhu, *Designing Embedded Systems with Arduino*, Springer, 2018.
3. Igoe, *Making Thinks Talk*, Maker Media, 2011.
4. Karvinen and Karvinen, *Make: Arduino Bots and Gadgets*, O'Reilly, 2011.

B. C Programming

5. Fiore, *Embedded Controllers Using C and Arduino*, 2019.
6. Purdum, *Beginning C for Arduino*, Apress, 2012.
7. Williams, *Make: AVR Programming*, O'Reilly, 2014.

Organization

Literature cont'd

C. Theory

8. Lee and Seshia, *Introduction to Embedded Systems*, MIT Press, 2017.
9. Buttazzo, *Hard Real-Time Computing Systems*, Springer, 2011.

Lectures: Slides will be available on Moodle.

Exercises: Students present solutions to problem sets.

Exam: Closed-book exam (90 min) at end of semester.

Syllabus

No	Date	Topic
1	10/2	Introduction
2	10/9	Facing the World
3	10/16	Sensors
4	10/23	Actuators
5	10/30	AVR Architecture
6	11/6	MCU System Architecture
7	11/13	C Programming
8	11/20	Assembly Programming
9	11/27	Timers and Interrupts
10	12/4	Communication
11	12/11	Real-Time Systems
12	12/18	Scheduling
13	1/8	Modeling
14	1/15	Model Checking

Introduction

Q: What is an Embedded System?

Answers:

1. An embedded system is a computer system, that is embedded in the physical environment to serve a specific purpose.

   Hopping robot: `bml.eecs.berkeley.edu`

   Flying machines: `raffaello.name/dynamic-works/`

2. The role of an embedded system is not to produce a final result but to maintain an ongoing interaction with its unpredictable environment.

   Theo Jansen's strandbeest: `www.strandbeest.com`

   Arduino controlled mini-strandbeest: `blog.arduino.cc/2017/11/09/mini-strandbeest-goes-electric-with-arduino/`

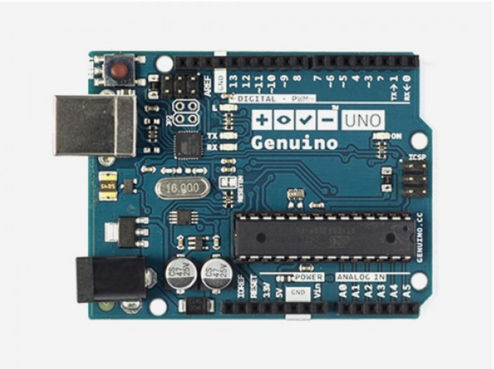
Requirements

Embedded systems are traditionally built with special-purpose hardware, in particular microcontroller units (MCU) rather than general-purpose architectures, because MCUs are tailored to meet the requirements of embedded computer systems:

- I/O capabilities: analog and digital interfaces
- Timing: time-triggered events and real-time capabilities
- Low-power: facilitate small form factors and long battery life
- Low-level programming: predictable control over hardware

Case Study: AVR

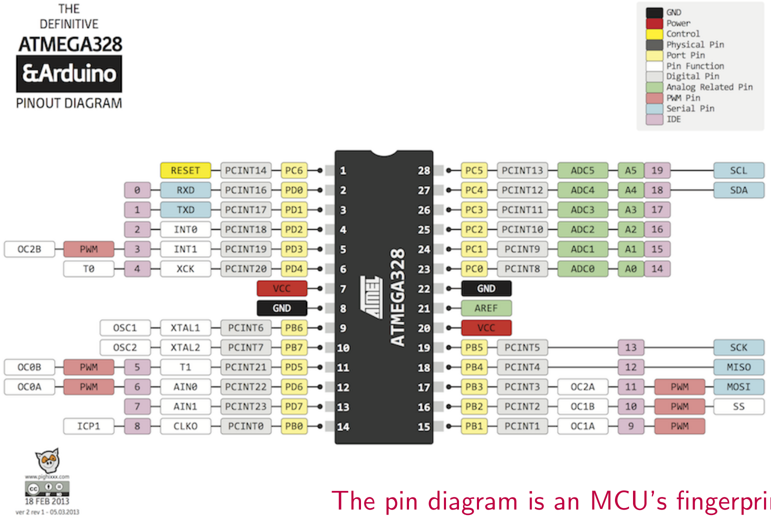
In this course, we learn about MCU architecture by studying Atmel's AVR (Advanced Virtual RISC). It is the MCU of choice in the DIY maker culture, and is the core of the Arduino technology.



AVR model ATmega328P, with dual-inline package (DIP) for easy replacement, on Genuino Uno board.

Case Study: AVR

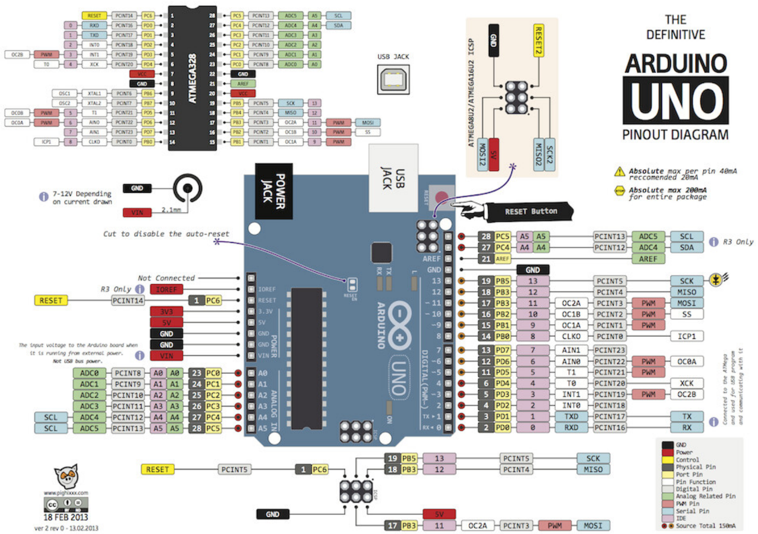
Pin diagram of the ATmega328P DIP package:



The pin diagram is an MCU's fingerprint.

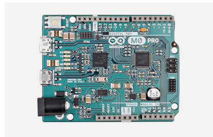
Case Study: AVR

Pin diagram of the Arduino Uno board:

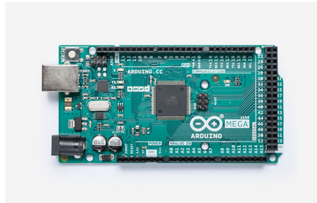


Alternative Development Boards

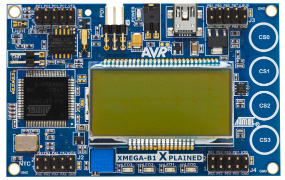
Arduino M0 PRO



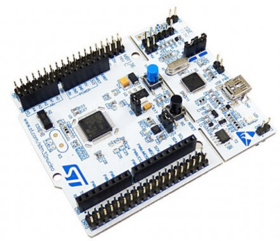
Arduino Mega



AT XMEGA



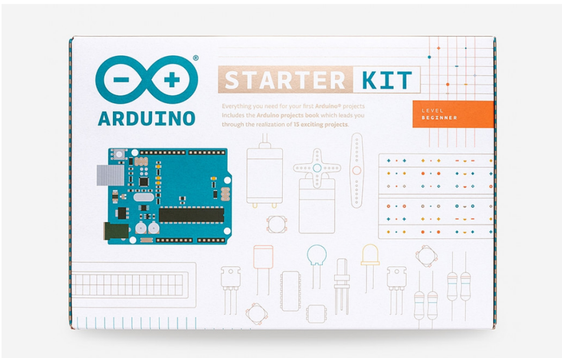
STM32F334



MSP430FR4133

Arduino Starter Kit

DIY: `store.arduino.cc/genuino-starter-kit`



Before taxes: 79.90 €

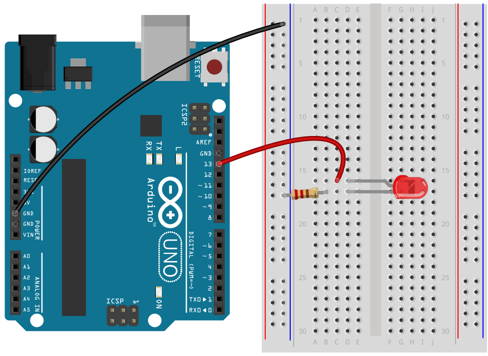
Learning Goals

- Understand the AVR microcontroller (Arduino)
- AVR programming in C and assembly
- Sensors and actuators
- Low-power design
- Real-time computing (freeRTOS)
- Model-based design

Embedded Systems 101

An actuator is a device that alters a physical quantity.

A light-emitting diode (LED) is an actuator that emits light if a current flows through it. An Arduino can blink an LED with this circuit



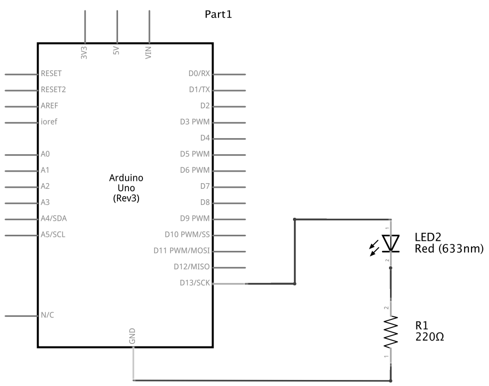
under control of the Arduino sketch (C program) on the next slide.

Embedded Systems 101

```
int state = 0;

void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  if (state == 0) {
    state = 1;
    digitalWrite(13, HIGH);  // LED on
  } else /* state == 1 */ {
    state = 0;
    digitalWrite(13, LOW);   // LED off
  }

  delay(1000);               // wait 1000ms = 1s
}
```


Embedded Systems 101

Q: How does the electrical circuit work?

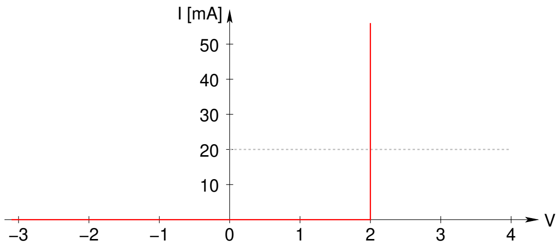


A: We model the functionality of the LED.

Embedded Systems 101

Digital model: An LED has the IV-characteristics of a *pn*-junction diode. The materials responsible for emitting light of a particular color affect the forward-bias voltage, $V_B \approx 2\,\mathrm{V}$ for a red LED:

$$\text{diode switch} = \begin{cases} \text{closed}, & V_{LED} \geq V_B\,, \\ \text{open}, & V_{LED} < V_B \end{cases}$$



AVR pins are designed for $I_{pin} = 20\,\mathrm{mA}$ output current, and 40 mA max or risking damage to the AVR chip.

Embedded Systems 101

The resistor is needed to limit current I_{pin} through the circuit:

$$V_{pin} = V_{LED} + R\, I_{pin}$$

Model the AVR pin as a voltage source:

$$V_{pin} = \begin{cases} 5\,\mathrm{V}\,, & \mathrm{digitalWrite}(pin,\ \mathrm{HIGH})\,, \\ 0\,\mathrm{V}\,, & \mathrm{digitalWrite}(pin,\ \mathrm{LOW})\,. \end{cases}$$

We must choose R s.t.

$$I_{pin} = \frac{V_{pin} - V_{LED}}{R} \approx 20\,\mathrm{mA}$$

or

$$R \approx \frac{5\,\mathrm{V} - 2\,\mathrm{V}}{20\,\mathrm{mA}} = 150\,\Omega$$

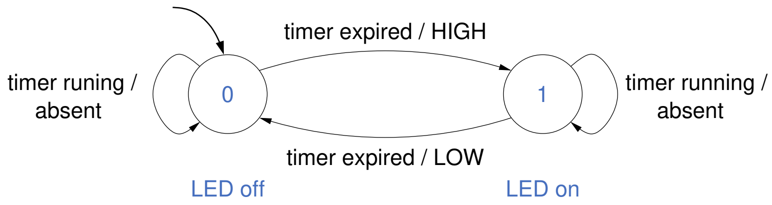
The next larger available resistor is $R = 220\,\Omega$ s.t.

$$I_{pin} = 13.6\,\mathrm{mA} \quad \text{and} \quad V_R = 3\,\mathrm{V}$$

Embedded Systems 101

Q: Why does the Arduino sketch blink the LED?

A: We model the controller program by means of an FSM.

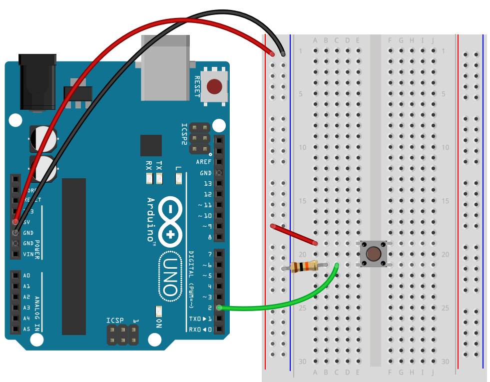


Analysis: The LED blinks, i.e. it toggles between on and off for the duration of the timer delay, respectively.

Embedded Systems 101

A sensor is a device that measures a physical quantity.

An AVR pin can be configured to act as a digital voltage sensor, i.e. to sense high ($\approx 5\,\mathrm{V}$) or low ($\approx 0\,\mathrm{V}$) voltage values. This circuit uses a push button to generate high or low voltages, and pin 2 as a sensor:



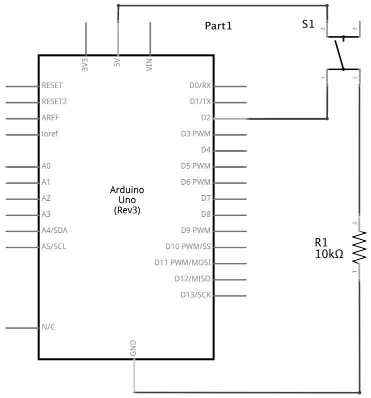
Embedded Systems 101

Arduino sketch: Sense voltage on pin 2, and send the digital voltage value, 0 or 1, via the serial port to the host computer.

```
void setup() {
  pinMode(2, INPUT);

  Serial.begin(9600);       // open serial port to host
}

void loop() {
  int v = digitalRead(2);  // sense voltage at pin 2
  Serial.println(v);       // send voltage to host

  delay(100);              // wait 100ms
}
```

Embedded Systems 101

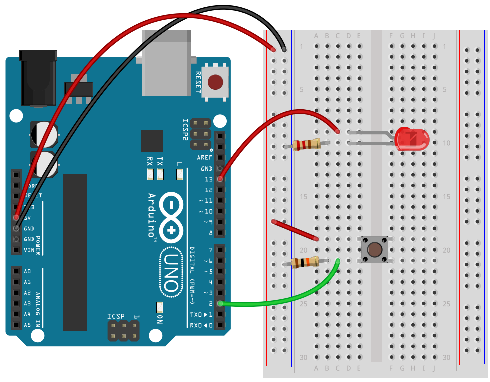
Schematic of electrical circuit:



$$V_{\mathrm{D2}} = V_{\mathrm{R1}} = \begin{cases} 5\,\mathrm{V}, & \text{S1 closed}, \\ 0\,\mathrm{V}, & \text{S1 open} \end{cases}$$

Embedded Systems 101

Sense and actuate: Switch the LED on while the button is pushed.



Polling strategy: Sample pushbutton position on pin 2 inside sketch `loop`, and switch on the LED at pin 13 while the button is pushed.

Embedded Systems 101

```
void setup() {
  pinMode(2, INPUT);          // sensor pin
  pinMode(13, OUTPUT);        // LED (actuator) pin
}

void loop() {
  int b = digitalRead(2);     // sense pushbutton position

  if (b == 1)                 // if button is pushed
    digitalWrite(13, HIGH);   //    LED on
  else                        // else
    digitalWrite(13, LOW);    //    LED off

  delay(1);                   // minimal delay
}
```


Embedded Systems 101

Pros of polling strategy:

- Responsiveness: The smaller the delay, the faster the MCU reacts to changes in the pushbutton position.
- Prioritization: In the presence of multiple sensors, the polling order determines the priority of the senors: poll most important first.

Cons of polling strategy:

- Power consumption: The smaller the delay the more power the MCU consumes.
- Power-time tradeoff: The choice of the delay interval is a tradeoff. The smaller the response time (good), the larger the power consumption of the MCU (bad), and vice versa.

Embedded Systems 101

Alternative strategy: interrupts wake up the MCU when a specified event occurs, and invoke the associated interrupt handler.

```
volatile byte state = LOW;     // initially button not pushed

void setup() {
  pinMode(2, INPUT);           // interrupt (sensor) pin
  pinMode(13, OUTPUT);         // LED (actuator) pin

  attachInterrupt(digitalPinToInterrupt(2),
                  handleButton, CHANGE);
}

void loop() {
  digitalWrite(13, state);
  delay(1);
}

/****** interrupt handler ******/
void handleButton() {
  state = !state;
}
```


Embedded Systems 101

Alternative strategy: interrupts wake up the MCU when a specified event occurs, and invoke the associated interrupt handler.

```
volatile byte state = LOW;     // initially button not pushed

void setup() {
  pinMode(2, INPUT);           // interrupt (sensor) pin
  pinMode(13, OUTPUT);         // LED (actuator) pin

  attachInterrupt(digitalPinToInterrupt(2),
                  handleButton, CHANGE);
}

void loop() {
  digitalWrite(13, state);
  delay(1);
}

/****** interrupt handler ******/
void handleButton() {
  state = !state;
}
```


Unreliable: need better pushbutton model (bounces).

Arduino Simulation

Free online simulator:

www.tinkercad.com/circuits

Features:

- Arduino Uno, incl. programming with sketches (or codeblocks)
- Various sensor and actuator models
- Easy prototyping before building a system in hardware
- Also: 3D CAD for creative makers

Caution: A simulator is as good as its models reflect reality. Models approximate reality only, subject to engineering tradeoffs.