# Security of the WWW

Yannick Chevalier
Université de Toulouse
CSA M1, Security

# Plan

# Why having Web Applications ?

## Server side

- Easy to change the frontend into an Android or iOS application

  client-server communication is quite different, but the resources exposed are similar

- Easy to reach more clients

  no need to download anything

- Interaction with other Web applications

  aggregation, references, ads

## Client side

- Everyone has a browser

  Differences between browsers are painful for frontend developers

- Fast prototyping

# General Architecture

## 3 parts

- A browser sends a request to a server
- A server receives requests and sends responses
- A `network` moves the messages to and from the client's and the server's computer

# Presentation

## Micro-controller

- AVR ATMega micro-controller
- “Harvard” architecture
- 8 bits native, 16 bits operations

## Board

In addition to a controller :

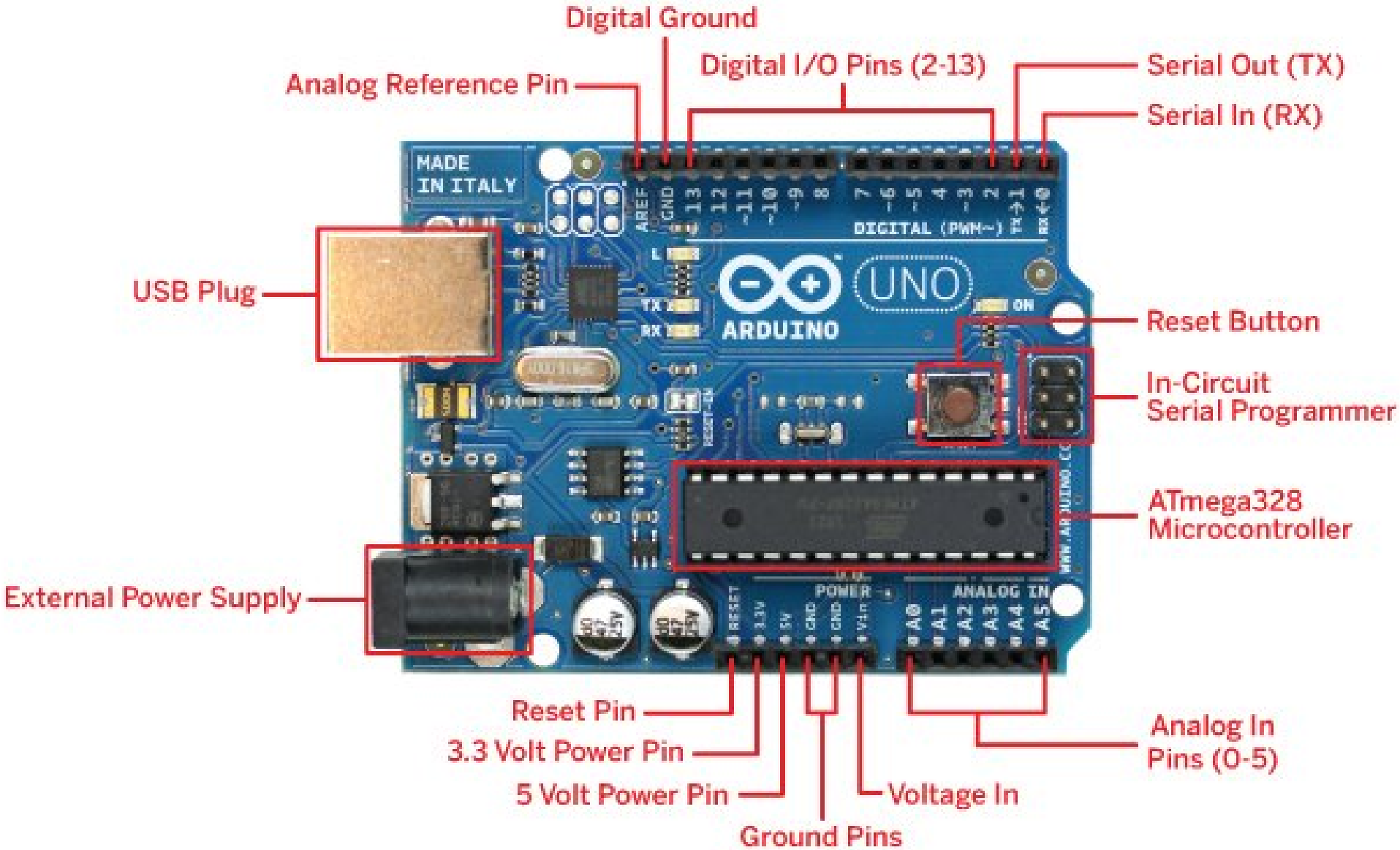
- Power and USB
- (Lot of) PINs to communicate with sensors and actuators

## Applications

- Basis for robotics, IoT, etc.
- Teaching embedded systems programming
- Fast & easy prototyping
- Ideal for low-cost DIY projects

high-end alternative : Raspberry $\pi$

# Plan


Digital Ground
Analog Reference Pin
Digital I/O Pins (2-13)
Serial Out (TX)
Serial In (RX)
USB Plug
Reset Button
In-Circuit
Serial Programmer
ATmega328
Microcontroller
External Power Supply
Reset Pin
3.3 Volt Power Pin
5 Volt Power Pin
Ground Pins
Voltage In
Analog In
Pins (0-5)
MADE
IN ITALY
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN

# Physical Security Analysis

1. List the physical areas and the possible communication channels

   Goal is to list communication channels with the internal of the $\mu$-controller when having access to the board

   $=$ Attack

   surface

2. Assess the expertise needed to actually use each of these channels

   Goal is to determine who can do what

3. Describe the attack surface without connecting directly to the PINs of the controllers

   Good compromise, most circuitry is electronics, not security

# Physical Security Analysis

1. List the physical areas and the possible communication channels

   Goal is to list communication channels with the internal of the $\mu$-controller when having access to the board

   $=$ Attack

   surface

2. Assess the expertise needed to actually use each of these channels

   Goal is to determine who can do what

3. Describe the attack surface without connecting directly to the PINs of the controllers

   Good compromise, most circuitry is electronics, not security

# Physical Security Analysis

1. List the physical areas and the possible communication channels

   Goal is to list communication channels with the internal of the $\mu$-controller when having access to the board

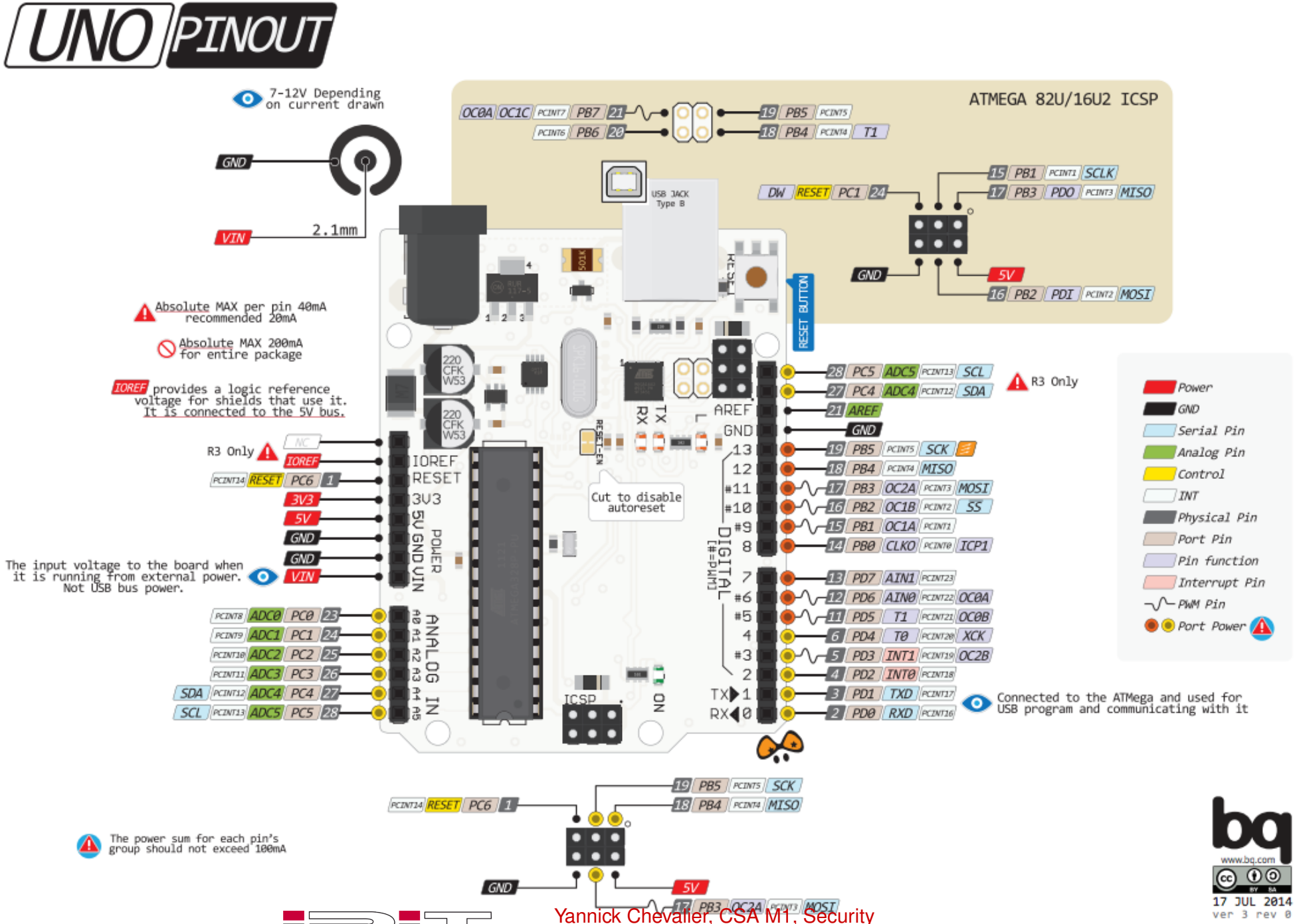
   $=$ Attack surface

2. Assess the expertise needed to actually use each of these channels

   Goal is to determine who can do what

3. Describe the attack surface without connecting directly to the PINs of the controllers

   Good compromise, most circuitry is electronics, not security

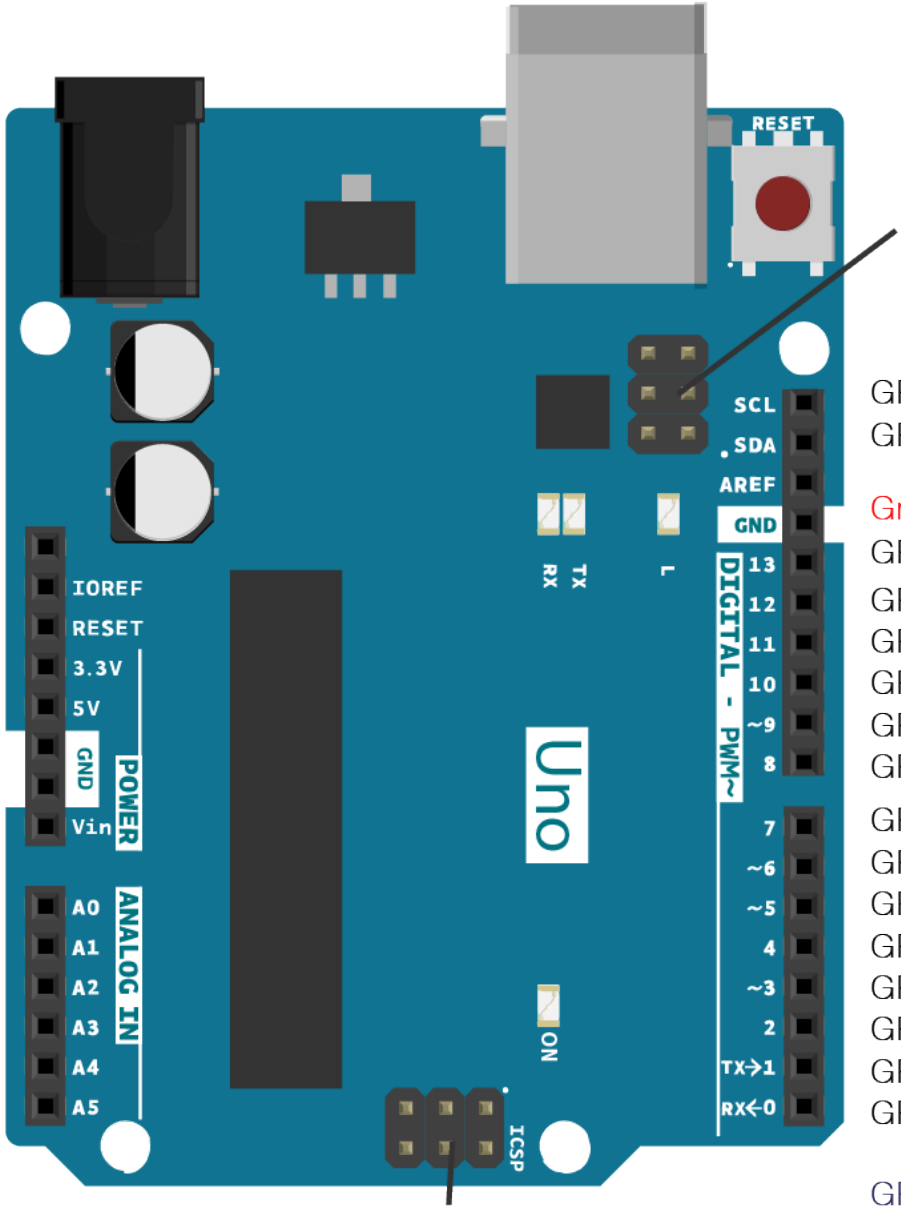

UNO PINOUT
7-12V Depending on current drawn
GND
VIN
2.1mm
ATMEGA 82U/16U2 ICSP
OC0A OC1C PCINT7 PB7 21
PCINT6 PB6 20
19 PB5 PCINT5
18 PB4 PCINT4 T1
USB JACK Type B
DW RESET PC1 24
15 PB1 PCINT1 SCLK
17 PB3 PDO PCINT3 MISO
GND
5V
16 PB2 PDI PCINT2 MOSI
RESET BUTTON
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
IOREF provides a logic reference voltage for shields that use it. It is connected to the 5V bus.
R3 Only
NC
IOREF
PCINT14 RESET PC6 1
3V3
5V
GND
GND
VIN
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
The input voltage to the board when it is running from external power. Not USB bus power.
PCINT8 ADC0 PC0 23
PCINT9 ADC1 PC1 24
PCINT10 ADC2 PC2 25
PCINT11 ADC3 PC3 26
SDA PCINT12 ADC4 PC4 27
SCL PCINT13 ADC5 PC5 28
ANALOG IN
A0 A1 A2 A3 A4 A5
220 CFK W53
220 CFK W53
TX
RX
RESET-EN
Cut to disable autoreset
AREF
GND
13
12
#11
#10
#9
8
7
#6
#5
4
#3
2
TX 1
RX 0
DIGITAL (PWM=~)
ICSP
ON
R3 Only
28 PC5 ADC5 PCINT13 SCL
27 PC4 ADC4 PCINT12 SDA
21 AREF
GND
19 PB5 PCINT5 SCK
18 PB4 PCINT4 MISO
17 PB3 OC2A PCINT3 MOSI
16 PB2 OC1B PCINT2 SS
15 PB1 OC1A PCINT1
14 PB0 CLKO PCINT0 ICP1
13 PD7 AIN1 PCINT23
12 PD6 AIN0 PCINT22 OC0A
11 PD5 T1 PCINT21 OC0B
6 PD4 T0 PCINT20 XCK
5 PD3 INT1 PCINT19 OC2B
4 PD2 INT0 PCINT18
3 PD1 TXD PCINT17
2 PD0 RXD PCINT16
Connected to the ATMega and used for USB program and communicating with it
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
19 PB5 PCINT5 SCK
18 PB4 PCINT4 MISO
PCINT14 RESET PC6 1
GND
5V
17 PB3 OC2A PCINT3 MOSI
The power sum for each pin's group should not exceed 100mA
bq
www.bq.com
17 JUL 2014
ver 3 rev 0

IORef: 5V

Vin: 7-12V DC max.

Serial: Serial is attached to pins 0 and 1, and to the USB-Serial micrcontroller on board.

The Uno has a second microcontroller on board to handle USB-to-serial communications.
This is the ICSP header for that microcontroller.




| Comm. | ADC | GPIO |
|---|---|---|
| | | IORef |
| | | Reset |
| | | +3.3V |
| | | +5V |
| | | Gnd |
| | | Gnd |
| | | Vin |
| | ADC0 | GPIO14 |
| | ADC1 | GPIO15 |
| | ADC2 | GPIO16 |
| | ADC3 | GPIO17 |
| SDA | ADC4 | GPIO18 |
| SCL | ADC5 | GPIO19 |

ICSP:

| | | |
|---|---|---|
| Reset | SCK | MISO |
| Gnd | MOSI | +5V |

| GPIO | ADC | Comm. | PWM | Interrupts | |
|---|---|---|---|---|---|
| GPIO18 | ADC4 | SDA | | | |
| GPIO19 | ADC5 | SCL | | | |
| | AREF | | | | |
| Gnd | | | | | |
| GPIO13 | | SCK | | | LED |
| GPIO12 | | MISO | | | |
| GPIO11 | | MOSI | PWM11 | | |
| GPIO10 | | CS | PWM10 | | |
| GPIO9 | | | PWM9 | | |
| GPIO8 | | | | | |
| GPIO7 | | | | | |
| GPIO6 | | | PWM6 | | |
| GPIO5 | | | PWM5 | | |
| GPIO4 | | | | | |
| GPIO3 | | | PWM3 | INT1 | |
| GPIO2 | | | | INT0 | |
| GPIO1 | | TX | | | |
| GPIO0 | | RX | | | |

# I/O Pins

## Most cases

- Can only retrieve or enter data
- RX/TX (reception/transmission) : UART protocol (for communication with a computer)
- MISO/MOSI/SCLK/NSS : SPI protocol, fast, but needs 4 pins
- SDA/SCL : I2C protocol, 2 pins needed for communication with a sensor/actuator or another Arduino
- Pins are linked to registers in the $\mu$-controller
- Analysis of the program in the $\mu$-controller :
  - Check what is done with data received (read on registers)
  - Check what data is sent on the pins (written on registers)

## Parallel Programming

- A special mode in which both the serial bus controller and the ATMega328 $\mu$-controller can be written
- More details later

# USB CONNECTOR

## MOST CASES

- Is actually employed to upload programs on the ATMega328 $\mu$-controller
- Controlled by the ATMega16U2 controller (USB interface), as well as the previous I2C and UART protocols

## KEYBOARD AND MOUSE

- The USB controller may pose as being connected to several USB devices
- Including a Mouse and a Keyboard in the standard library
- In that case it can send keyboard and mouse events
- This is used in rogue USB devices (*e.g.* USB keys) to penetrate a system

# In-Circuitry Serial Programming

## Usage

- Update to the serial controller and the ATMega328 $\mu$-controller
- These updates cannot be controlled
- As in Parallel Programming, these updates start with erasing all the memory

## Security Analysis

- Availability : can be ensured only if reset disabled
- Confidentiality is preserved in case of reset (all bits are really set to 1)
- Integrity :
  - Need to focus on the case of an apparently functioning device
  - Authentication : The device serial number is not sufficient to decide we can send information to this device, the program may have been replaced with a malware

# Parallel Programming

## Principle

- Uses 20 pins for power, data, and control
- Faster than serial

## Practical Aspect

- Mostly with a dedicated “writer” device set up to program other devices
- Same comments as for ICSP *re.* capabilities and reset

# SUMMARY

## SECURITY CONCERNS

- Sensitive data inside the $\mu$-controller
  - Programs integrity needs to be protected
  - Data integrity and confidentiality
- Attack surface :
  - The pins and the USB plug
  - Information on these pins is either :
    - "passive", and given as data in registers of the $\mu$-controller
    - "active", and can reset the $\mu$-controller without any recourse
  - In all cases, additional channel : interruptions telling that some data is available ($+$ internal clock interruptions)
- Next step : look at the internals of the $\mu$-controller to check the security of programs

# Plan

# Outline

## The ATMEL ATMega328P controller

- Memory Model
- Memory Security

# Harvard Architecture

## Von Neumann Architecture

- Unified address space space for data and programs
- Most common at the application level

  Windows and Unix processes have a unique address space

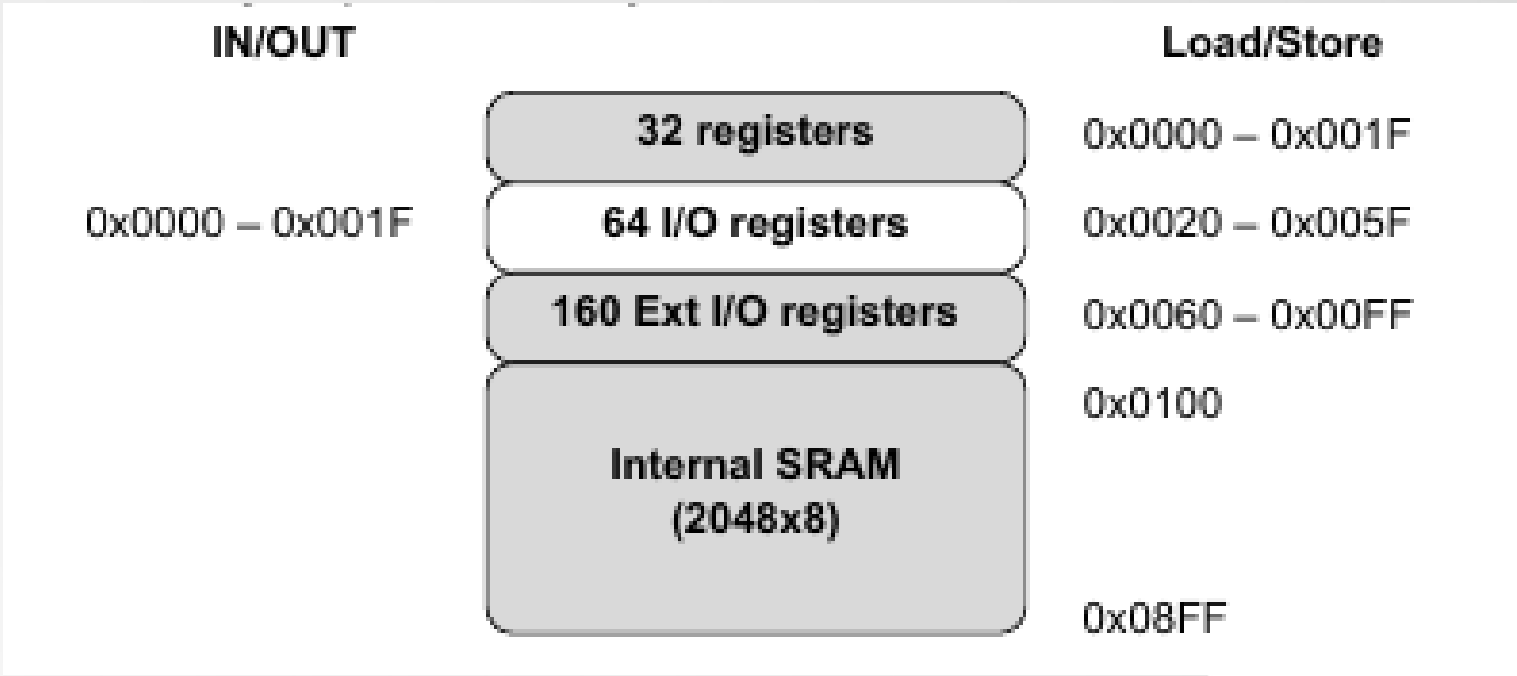
## Harvard Architecture (ATMega)

Three disjoint address spaces :

- Program memory (32kB) : program code and constant data
- SRAM (2kB) : the place where variables' values are stored during the execution of programs, including 16bits registers (L/H)
- EEPROM (1kB) : mostly for flags configuring the processor's functions and security
- The address `0x200` corresponds to two different bytes, one in the program memory, one in the SRAM

The address space to be used depends on the instruction (different load/store instructions)

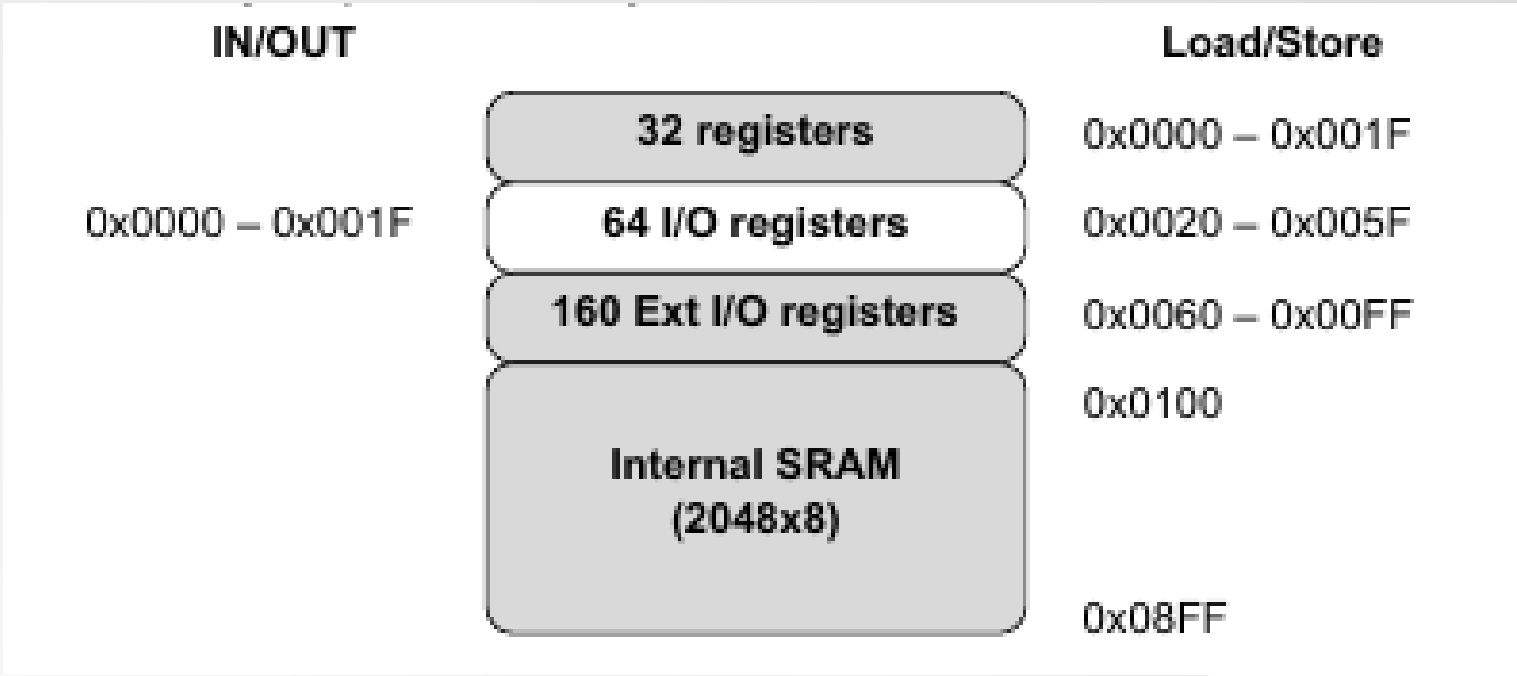
# THE SRAM




## 32 REGISTERS (CBI/SBI AND LD/ST)

- Employed for arithmetic, etc.
- They have names
- X, Y, Z : used for indirect addressing (functions, table of symbols)

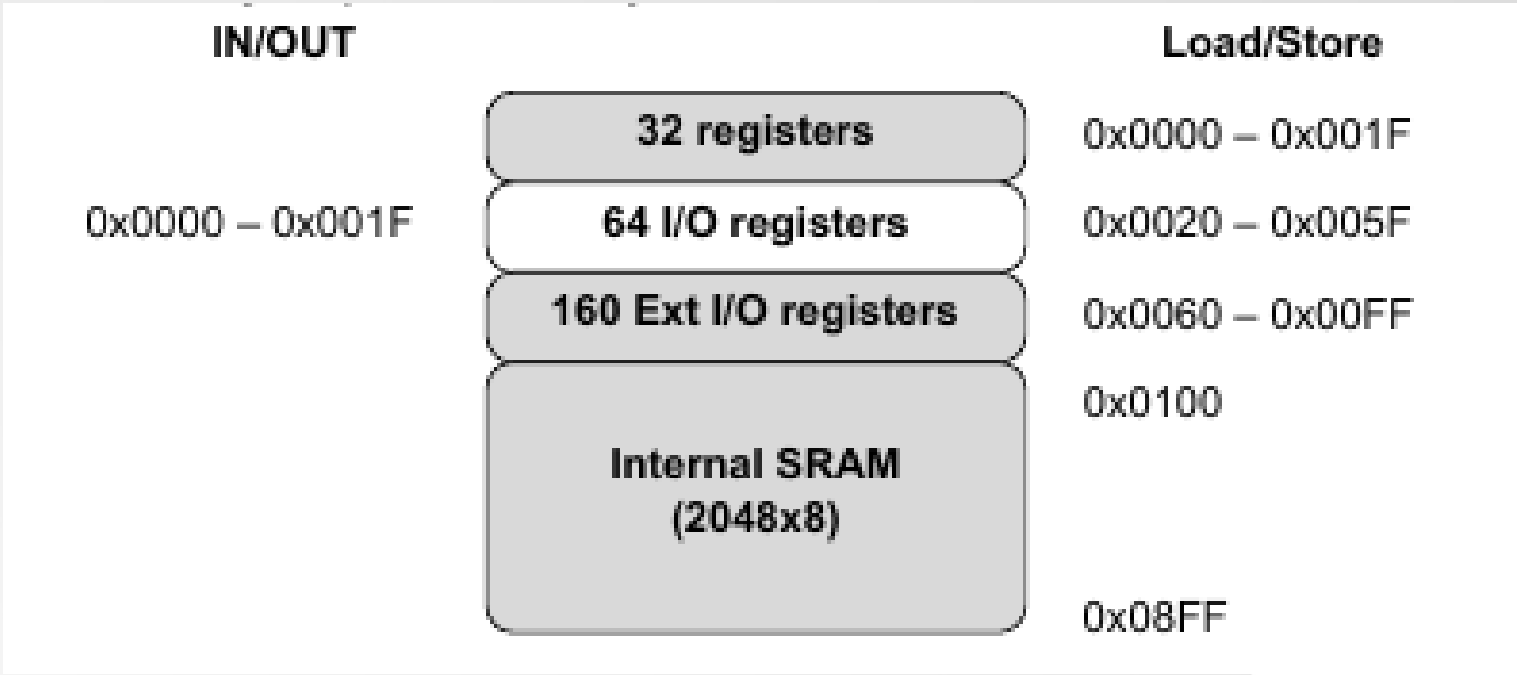
# THE SRAM




## 64 REGISTERS (IN/OUT AND LD/ST)

- Those connected to the “outside”
- They have names too
- Example : stack pointer is here

# The SRAM




## Other registers & memory (LD/ST)

- Most registers still have names
- Registers for communication protocols are here
- Only LD/ST can be used to read/write here
- Stack and heap are in the rest of the SRAM

# The EEPROM

## Instructions

- LD/ST (load/store) : take 1 address and 1 byte
- LD : copy the byte at that address to the target byte
- ST : copy the byte into the target address

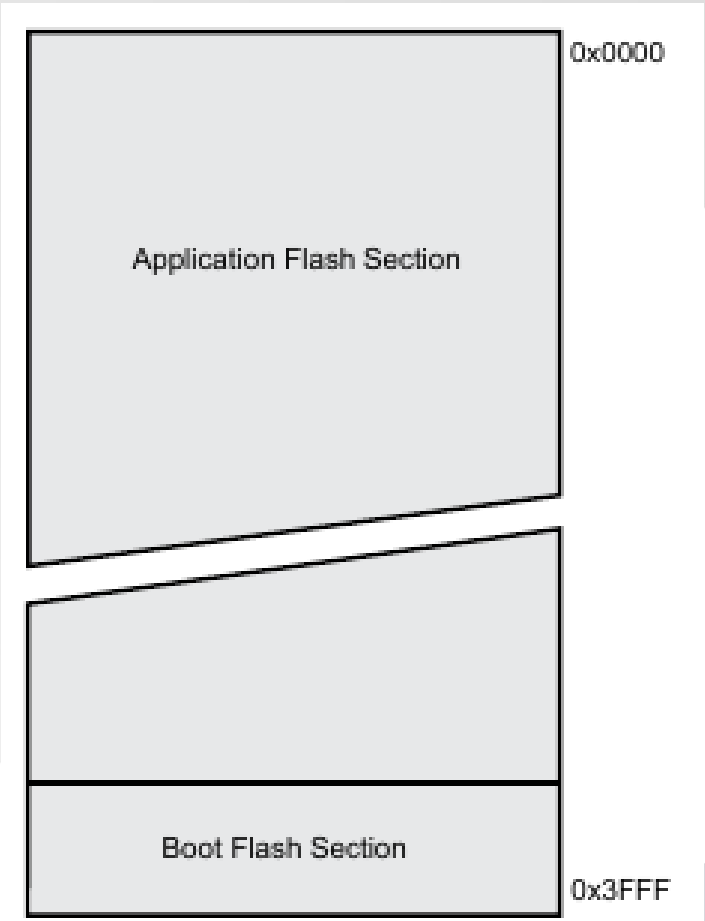
## LD/ST and the address space

- One 16b register at address `0x41-0x42` of the SRAM
- One 8b register at address `0x40` of the SRAM
- LD/ST with these registers will interact with the EEPROM instead of the SRAM

# The Program Memory

## Description

- Contains the instructions
- Two security levels : Application and BootLoader Space (BLS)
- Instructions are read by the processor
- Possible communication with the SRAM :
  - LPM : Load from address in program memory to address in SRAM
  - SPM : Store from address in SRAM to address in program memory



# Exercice

1. Make a diagram of the different memory parts, each with the communication channels to and from that part
2. How would you model them in the BLP/Biba models ?
3. What are the subjects and the objects here ? What can you say on Access Control ?

# Outline

## The ATMEL ATMega328P controller

- Memory Model
- Memory Security

# The LPM/SPM instructions

## LPM

- This instruction can appear anywhere in the Program Memory
- It is very useful to store strings in the program data, before loading them in the memory for further processing
- Its behaviour is controlled by lock bits

## SPM

- This instruction can only be in the Boot Flash Section
- When seen at an address in the Application Flash Section, it is replaced with a NOP
- Its behaviour is also controlled by lock bits

# Access Control on Physical Programming

| Memory Lock Bits | | | Protection Type |
|---|---|---|---|
| LB Mode | LB2 | LB1 | |
| 1 | 1 | 1 | No memory lock |
| 2 | 1 | 0 | No write with Parallel & Serial Programming |
| 3 | 0 | 0 | No read (verification)&write with Parallel & Serial Programming |

## Notes

- Lock bits are in EEPROM, by default (after reset) the value is one, setting a bit means giving it the value 0
- The (parallel or serial) programmer can still issue a chip erase command, that will unset the lock bits
- Conclusion : Availability cannot be ensured against physical attackers, but confidentiality and integrity can be preserved (against programmers)

# Access Control on Software Instructions (LPM/SPM)

| BLB0 Mode | BLB02 | BLB01 | Protection Type |
|---|---|---|---|
| 1 | 1 | 1 | No restriction for SPM or LPM access to the Application Section |
| 2 | 1 | 0 | SPM not allowed to write to the Application Section |
| 3 | 0 | 0 | above + below |
| 4 | 0 | 1 | LPM in the BLS not allowed to read the Application Section, interruptions disabled if stored in the BLS while executing the Application Section |

## Notes

- What is the security model here ?
- Why are interruptions disabled when they are stored in the BLS ?

# Access Control on Software Instructions (LPM/SPM)

| BLB1 Mode | BLB12 | BLB11 | Protection Type |
|---|---|---|---|
| 1 | 1 | 1 | No restriction for SPM or LPM access to the BLS |
| 2 | 1 | 0 | SPM not allowed to write to the BLS |
| 3 | 0 | 0 | above + below |
| 4 | 0 | 1 | LPM in the Application Section not allowed to read the BLS, interruptions disabled if stored in the Application Section while executing the BLS |

## Notes

- What is the security model here ?
- Why are interruptions disabled when they are stored in the Application Section ?

# Plan

# Conclusion&Moral

## Access Control

- Fine grained access control is possible
- Availability cannot be ensured, but Confidentiality& Integrity can be preserved

## Authentication

- By default, an identifying serial number is available, but provides no guarantee against Chip Erase and complete reprogramming

  Authentification of the physical device or of its software ?

- Real authentication with cryptography is possible
- Keys can be kept confidential unless against a very strong attacker

## Moral

- As promised, even the old BLP and Biba models inform the design of current systems
- Functionality *vs* Security : “hot” updates (with SPM) are very desirable, but needs to trust the bootloader section