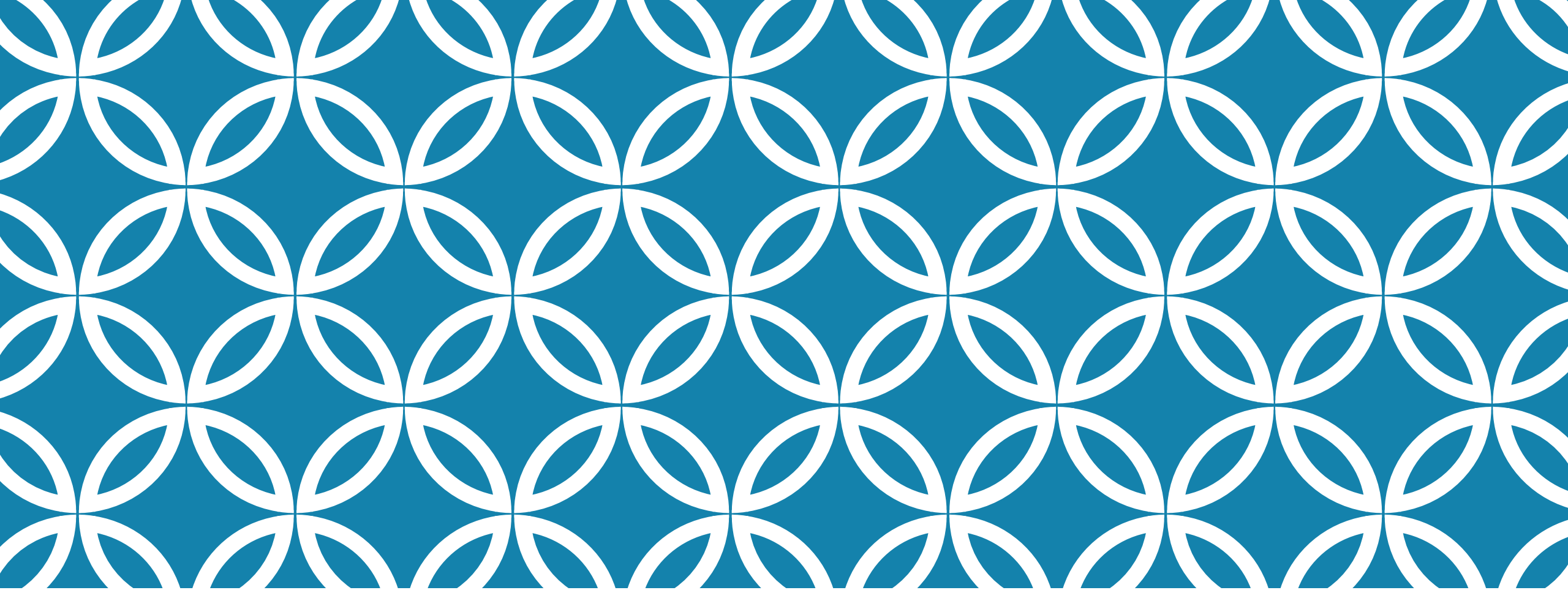

THE ARDUINO PLATFORM AND C PROGRAMMING

ZIED JARRAYA
SAIFEDDINE BARKIA
MOHAMED AZIZ TOUSLI

ARDUINO PLATFORM

- 8- bit microcontroller
- ATmega328: µP of Arduino
- ATmega16U2: µP of USB communication
- IDE: Integrated Development Environment
- Development Board
- Shields
- Firmware: Preprogram = Program written already (Unmodifiable code)
- Bootloader: Firmware of Arduino
- ICSP (In-Circuit Serial Programming): Method to program Bootloader
- Schematic: Wiring diagram of a circuit

ARDUINO PLATFORM

- Flash memory: Non volatile memory
- SRAM memory: Volatile memory (Power off = Bye)

Serial monitor: To communicate with Arduino

Code → Combine & Transform → Compile → Link (+Libraries) → Hex File Creation (.hex) → Upload on Arduino

PS: avr –gcc #Cross compilation: Creates an executable object file .o that works on another machine (avr for ATmega)

Sketch = Program

DIGITAL VS ANALOG

Digital	Analog
Integers	Continuous
0 or 5V	0 to 5V (0 to 1023)
R / W	R (ADC)
0-13 pins	A0-A5 pins

✓PS: Arduino doesn't contain analog output (No DAC)

IDE

void setup() #Initialization (Executed 1 time)

void loop() #While power is on, loop

pinMode(pin,MODE) #Setup, MODE=IINPUT;OUTPUT;INPUT_PULLUP

digitalRead(pin) #returns 0 or 1

digitalWrite(pin,VALUE) #VALUE=HIGH;LOW / 0;1

analogRead(analogPin) #returns 0 to 1023 (integers) ⇔ 0 to 5V

PS: INPUT_PULLUP: INPUT + Reverse polarity

DEBUGGING

- Controllability: Inputs / Internal memory
- Observability: Outputs / Internal memory

  ➔ Testing & Debugging

- Run control of the forget (stop)
- Real time monitoring of target execution
- Timing & functional accuracy

Debugging

- **Remote debugger** (debug monitor, maintains communication link #break;) → Software
- **Embedded debug interface** (in processor) → Hardware

Serial protocols (communication protocol between devices to debug with in Arduino, UART)

UART

UART (Universal Asynchronous Receiver/Transmitter)

→ Used by modems to communicate with network

Parallel IN → **Tx** (Serialized) → Serial OUT = Serial IN → **Rx** (Deserialized) → Parallel OUT

Buffer Transmit/Receive: Table that groups everything transmitted/received

Baud rate: Number of transition per second $f = \frac{1}{T}$ (in bps: bauds per second)

UART

- Start & Stop bits have a specific duration to be recognized ($\geq \frac{T_{baud}}{2}$)
- Bit duration: T_{baud} : Each bit is sent in a fixed duration
- The duration must be know to Tx and Rx
- Transmission rate is less than baud rate
- Parity bit: Bit to check errors
  - Even parity: Number of 1's is even
  - Odd parity: Number of 1's is odd

PS: Error → Retransmission

Transmission efficiency: $\frac{8}{11} = 73\%$ (Send 11 bits to really send 8 bits)

Data throughput=Transmission rate=baud rate * transmission efficiency

SERIAL COMMUNICATION

UART: Protocol Arduino – USB (Debugging with serial monitor)

Serial.begin(baudrate#9600,#config) #Setup

Serial.print("text"/code ASCII) #Write from Arduino to serial monitor

Serial.write(value)

Serial.available() #How many bytes are waiting in the buffer #while (Serial.available() == 0);

Serial.readBytes() #Reads bytes from the buffer

Serial.read() #Read from serial monitor

PS: -1 if no data is available