Sketching with Hardware

06: Programming

Programming the Arduino

- Microcontrollers are restricted in memory and computing power
- Programs are written in C or C++
- Rather simple applications → simple programs
- Different patterns and paradigms than e.g. in Java

Basic Program Structure

```
// declare variables here

// the setup routine runs once on startup
void setup() {

}

// the loop routine runs over and over again forever:
void loop() {

}
```

Arduino-specific Functions

Function	Description
`delay(int micros)`	Delays the program by *`micros`* microseconds
`pinMode(int pin, int direction)`	Defines a pin as *`INPUT`* or *`OUTPUT`*. Should be called during *`setup()`*.
`digitalRead(int pin)`	Reads value from a pin. Returns *`HIGH`* (1) or *`LOW`* (0).
`analogRead(int pin)`	Reads analog value from a pin. Returns 0 to 255.
`digitalWrite(int pin, int value)`	Sets a pin to either *`HIGH`* (5 V) or *`LOW`* (0 V).
`analogWrite(int pin, int value)`	Sets voltage on a pin to a value between 0 (0 V) and 255 (5 V). Only works with pins that support *PWM*.
`Serial.println("Hello")`	Writes to the serial monitor and can be used for debugging. Call *`Serial.begin(9600)`* during *`setup()`*.

Non-blocking code

- The Arduino can only run one thread at a time
- No parallelization of tasks
- The `delay()`-function blocks the whole program
- `delay()` can be avoided by using a timer

Simple Timer

```
int buttonPin = 5;
int ledPin = 13;
int ledState = LOW;

void loop() {
  // blink the LED
  if(ledState == LOW) {
    ledState = HIGH;
  }
  else {
    ledState = LOW;
  }

  // turn off LED if button is not pressed
  if(digitalRead(buttonPin) == LOW) {
    ledState = LOW;
  }

  digitalWrite(ledPin, ledState);
  delay(1000); // wait for a second
}
```

blocking

```
// ...
long lastMillis = 0;

void loop() {
  // check time since last update
  if(millis() - lastMillis >= 1000) {
    lastMillis = millis();

    if(ledState == LOW) {
      ledState = HIGH;
    }
    else {
      ledState = LOW;
    }
  }

  // turn off LED if button is not pressed
  if(digitalRead(buttonPin) == LOW) {
    ledState = LOW;
  }

  digitalWrite(ledPin, ledState);
}
```

non-blocking

Code Structure

- Arduino code is often simple and sequential
- Tends to get messy pretty fast
- Debugging code and hardware at the same time can be tedious
- Well-structured code leads to better readability and easier debugging

State Machine: Cookie Dispenser

```
#define WAIT 1
#define ORDER 2
#define PAYMENT 3
#define DISPENSE 4

int state = WAIT;
int order;

// cookie dispenser functions
int getInput() { … }
int getOrder() { … }
void handlePayment(int product) { … }
void dispenseProduct(int product) { … }
```

```
void loop() {
  switch(state) {
    case WAIT:
      if(getInput() != 0) state = ORDER;
      else delay(1000);
      break;
    case ORDER:
      order = getOrder();
      state = PAYMENT;
      break;
    case PAYMENT:
      handlePayment(order);
      state = DISPENSE;
      break;
    case DISPENSE:
      dispenseProduct(order);
      state = WAIT;
      break;
}
```

Debug Levels

- The Arduino IDE has no builtin debugger
- Serial output is used to log the state of programs
- Using debug levels lets you switch between different levels of verbosity quickly

```
#define NONE 0
#define ERROR 1
#define WARN 2
#define DEBUG 3
#define ALL 4
#define DEBUG_LEVEL WARN

void loop() {
  if(DEBUG_LEVEL >= WARN) {
    Serial.println(“Warning!”);
  }
}
```