Extending mbeddr with Arduino support

Kolja Dummann

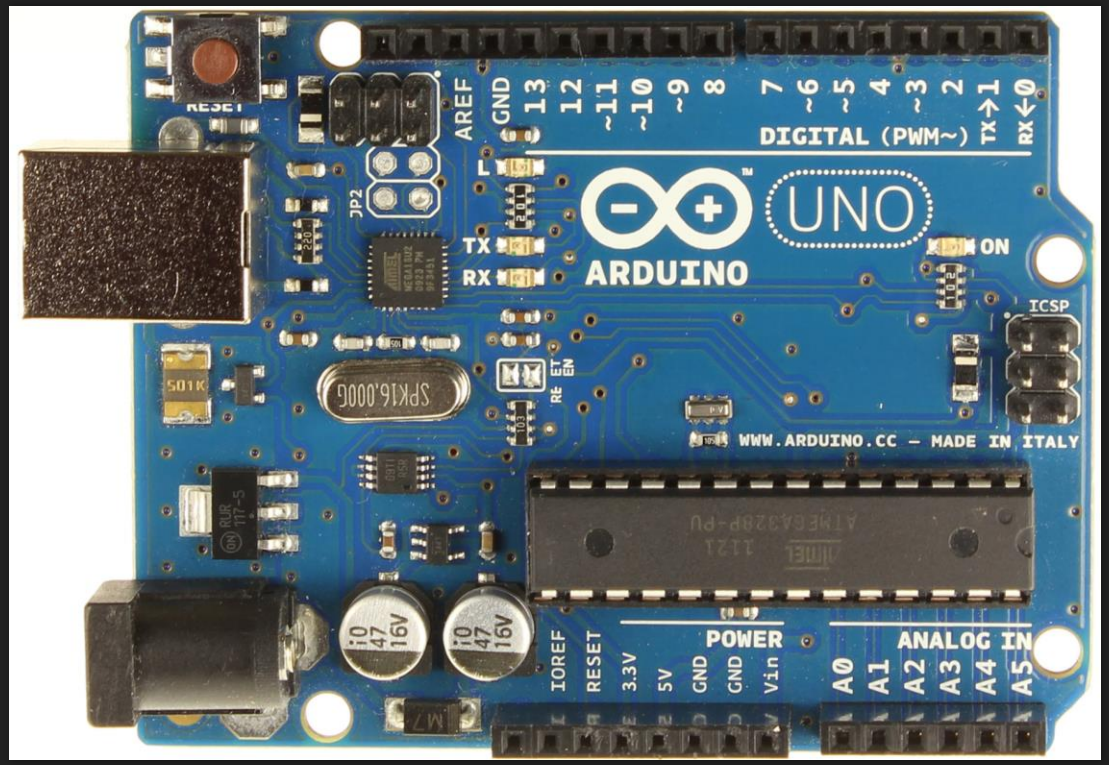
http://logv.ws

@dumdidum

Provide extensible first class language concepts for hardware interaction and description.

Arduino

“Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments."



Why Arduino?

open source software **and** hardware

lots of hardware extension

many compatible boards

huge community

mbeddr

“mbeddr supports embedded software development based on an extensible version of the C programming language and an IDE”

Why mbeddr?

open source

extensible C implementation

based on JetBrains MPS

awesome user guide and documentation

What's working?

Hardware and platform description

```
Arduino Platform Description
============================
name : uno
CPU speed : 16 MHz
MCU : atmega328
status register adress: 95
ADC Multiplexer Selection Register 124
ADC Control and Status Register A 122
ADC Low Register 120
ADC High Register 121

registers :
  8 bit register
    name = PINB
    address = 35
  8 bit register
    name = DDRB
    address = 36
```

Project configuration

```
Build System:
  arduino uno

Configuration Items
  pin configuration :
     digital :
     pin 0 name = doorLock configuration = input
     pin 1 name = lazerBeam configuration = output
     pin 2 name = fluxCompensator configuration = output
     pin 3 name = digitalPin3 configuration = none
     pin 4 name = digitalPin4 configuration = input
     pin 5 name = digitalPin5 configuration = input
     pin 6 name = digitalPin6 configuration = output
     pin 7 name = digitalPin7 configuration = output
     pin 8 name = digitalPin8 configuration = none
     pin 9 name = digitalPin9 configuration = none
     pin 10 name = digitalPin10 configuration = none
     pin 11 name = digitalPin11 configuration = none
     pin 12 name = digitalPin12 configuration = none
     pin 13 name = doorSwitch configuration = output
     analog :
     pin 0 name =  lightSensor
     pin 1 name =  analogPin1
     pin 2 name =  analogPin2
     pin 3 name =  analogPin3
     pin 4 name =  analogPin4
     pin 5 name =  analogPin5
```

Makefile generation

```
1  CC=avr-gcc
2  CFLAGS=-Os
3  OBJCOPY=avr-objcopy -O ihex -R .eeprom
4  ODIR=./bin
5  _OBJ_arduino=main.o
6  OBJ_arduino=$(patsubst %,$(ODIR)/%,$(_OBJ_arduino))
7
8
9  all: removeStuffFromLibraries clean arduino.hex
10 .PHONY: removeStuffFromLibraries all clean
11 removeStuffFromLibraries:
12
13 $(ODIR)/%.o: %.c
14                 mkdir -p $(ODIR)
15         $(CC) $(CFLAGS) -mmcu=atmega328 -DF_CPU=16000000UL     -c -o $@ $<
16 arduino: $(OBJ_arduino)
17         $(CC) $(CFLAGS)  -mmcu=atmega328 -o $@ $^
18 clean:
19         rm -rf $(ODIR)
20 arduino.hex : arduino
21         $(OBJCOPY) $< $@
```

Digital I/O + PWM

```
[pin initializer]
exported int8 main() {
  boolean dummy = true;
  while ( dummy ) {

    doorSwitch = high;
    delay( 500 );
    doorSwitch = low;
    delay( 500 );

  } while
```

Analog inputs

```
if (lightSensor == 50) {

} if
```

Interrupts

```
int16 foo = 0;
atomic {
   foo++;
   foo = foo * 2;
}
```

```
interrupt-driven instance protocol
statemachine ProtocolSM initial = stby {
  in msgReceived() interrupt 12
  var int8 sessionID = 0
  state stby {
    on msgReceived [ ] -> receiving { sessionID = someMemoryAccessAPI()[0]; }
  } state stby
  state receiving {

  } state receiving
}
```

What's next?

EEPROM support

```
memory layout {
   region ram: 0..1024
   region eprom: startOf(ram)..2048
   region devices: endOf(eprom)..startOf(devices) + sizeOf(ram) * 2
}
```

Better IDE integration

Upload code from mbeddr

On device debugging

Integrated run/debug configuration

Extended sensor and shield support

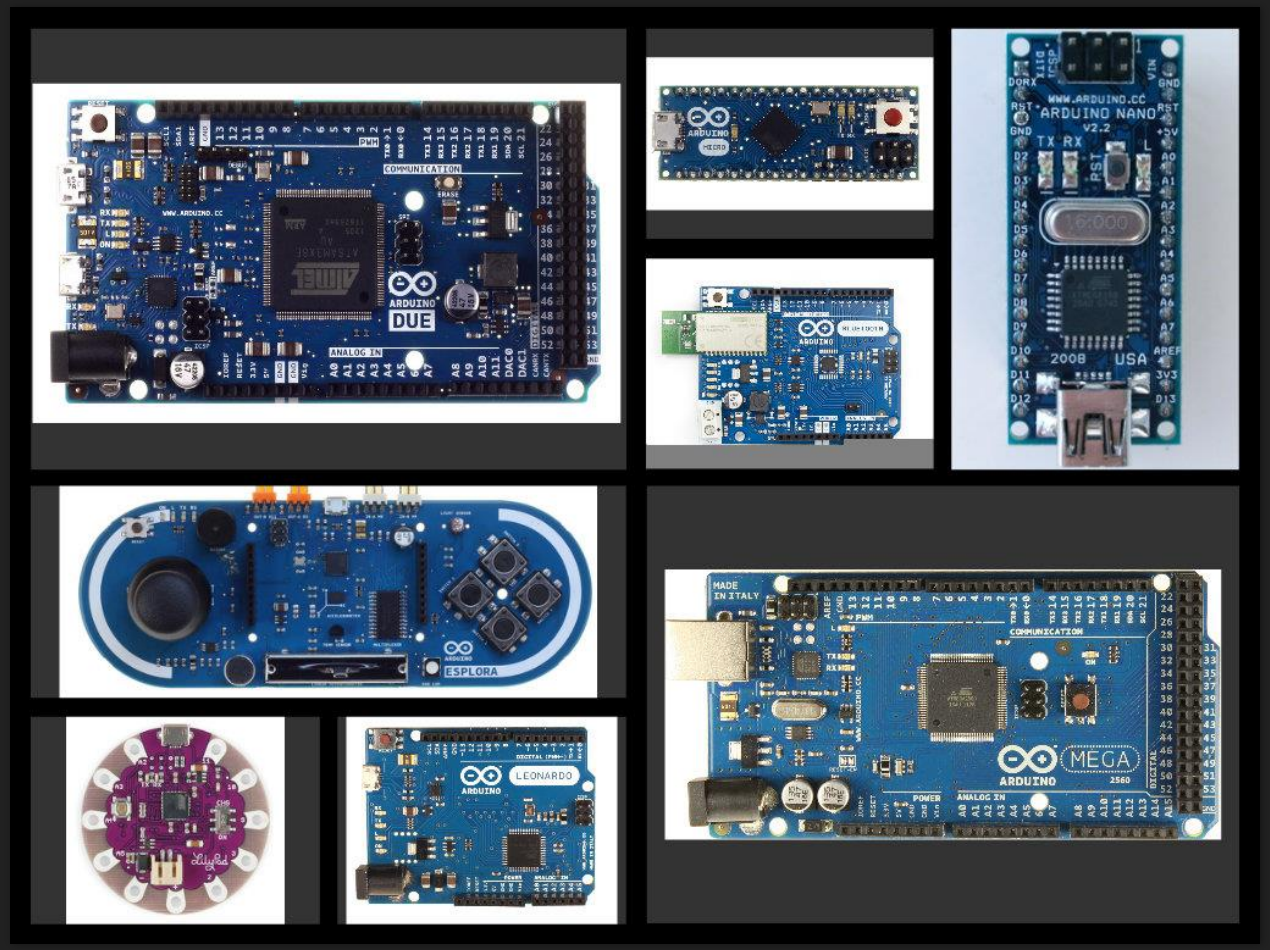
sensors:

- Temperature (PT 1X, NTC)
- Light
- Vibration
- …

shields:

- Ethernet
- Bluetooth
- NFC / RFID
- …

More boards



All of it is Open Source

Eclipse Public License
https://github.com/coolya/mbeddr.arduino/