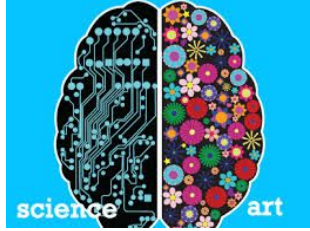


The Art & Science of Embedded Systems

Theoretical and practical session

MORMYTRON

Intro

Mormytron Labs

- FW consultancy company
- Activities
  - On-site FW development
  - Workshops
  - Product development
- Areas of expertise
  - Medical
  - Wearables
  - Aviation

Brain implant

X-ray machine

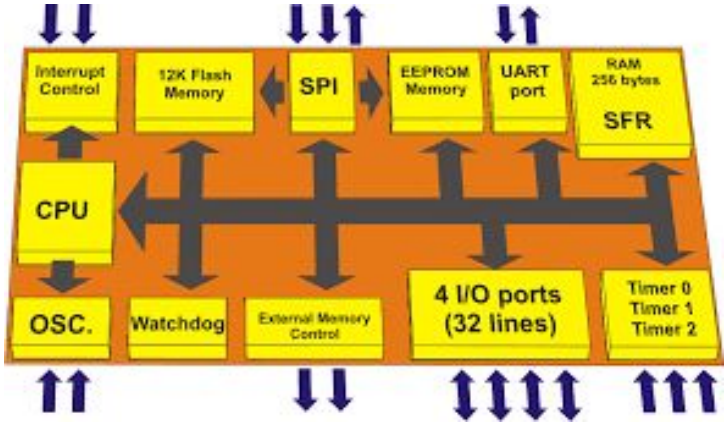
Smartwatch

Pregnancy monitor

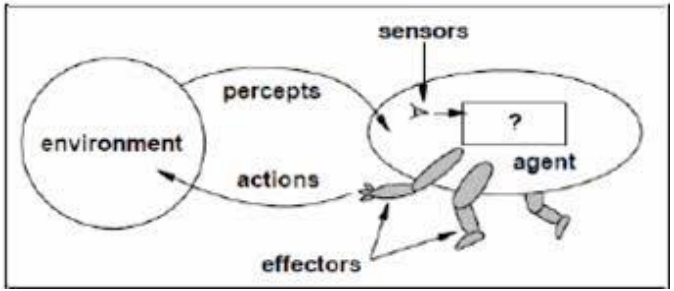
C/C++ language for embedded systems

- Why C/C++?
  - Compiled:
    - runs on native hardware
    - no need for runtime/interpreter (unlike Python, Java)
  - Typed: full control over memory footprint
  - Popular: forums and books provide knowledge
- Ideal for embedded projects
  - Tool vendors 99% of time provide C/C++ API
  - Compilers optimized for embedded code (memory usage vs performance)
  - Large amount of free libraries (code re-use speeds up development)
- Scalable, can tackle hugely complex projects
  - Bare metal (no OS, wait for events in a loop)
  - Scheduler (tick based)
  - Real Time OS

Microcontrollers (MCU)



- What is it?
  - Device with interface to the real world
  - Digital, analog I/O
  - Peripherals
  - Supporting hardware: counter, Real Time Clock, ADC, …
  - Communication buses: talking to other devices / sensors
    - I2C
    - UART
    - SPI
- Sensors and effectors can be attached
  - Sensor measures the state of the environment
  - Effectors change the environment



Basic programming concepts - overview

Source code (sketch): textual, human readable set of instructions (program)

Compile (verify): make machine executable code from source code

Variable: store/remember a number

Array: set of variables of the same type, no need to name them individually

Function: stop, execute a piece of code, then go back where it left off

Delay: do nothing for some time, then continue program execution

Basic programming concepts - syntax

Semicolon

Types

Expressions

Comments

More info: https://www.tutorialspoint.com/cprogramming/c_basic_syntax.htm

Basic programming concepts - control structures

If

Else

Switch

Case

https://www.tutorialspoint.com/cprogramming/c_decision_making.htm

Basic programming concepts - loops

For

While

https://www.tutorialspoint.com/cprogramming/c_loops.htm

Basic programming concepts - arrays and pointers

Pointer vs array

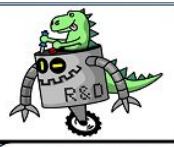
Underlying storage

Address in memory

https://www.tutorialspoint.com/cprogramming/c_arrays.htm

Arduino Cheat Sheet v.02b

Mostly taken from the extended reference:
http://arduino.cc/en/Reference/Extended
Gavin Smith – Robots and Dinosaurs, The Sydney Hackspace



Structure
```
void setup()   void loop()
```

Control Structures
```
if (x<5){ } else { }
switch (myvar) {
  case 1:
    break;
  case 2:
    break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } while (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto          // considered harmful :-)
```

Further Syntax
```
// (single line comment)
/* (multi-line comment) */
#define DOZEN 12 //Not baker's!
#include <avr/pgmspace.h>
```

General Operators
```
= (assignment operator)
+ (addition)        - (subtraction)
* (multiplication)  / (division)
% (modulo)
== (equal to)       != (not equal to)
< (less than)       > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and)     || (or)      ! (not)
```

Pointer Access
```
& reference operator
* dereference operator
```

Bitwise Operators
```
& (bitwise and)  | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left)  >> (bitshift right)
```

Compound Operators
```
++ (increment)    -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)
```

Constants
```
HIGH | LOW
INPUT | OUTPUT
true | false
143 // Decimal number
0173 // Octal number
B11011111 //Binary (8-bits only)
0x7B // Hex number
7U // Force unsigned
10L // Force long
15UL // Force long unsigned
10.0 // Forces floating point
2.4e5 // 245,000
```

Data Types
```
void
boolean          (0, 1, false, true)
char   (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte   (0 to 255)
int    (-32,768 to 32,767)
unsigned int (0 to 65535)
word   (0 to 65535)
long   (-2,147,483,648 to
        2,147,483,647)
unsigned long (0 to 4,294,967,295)
float  (-3.4028235E+38 to
        3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes
```

Strings
```
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
  //Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";
```

Arrays
```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
```

Conversion
```
char()          byte()
int()           word()
long()          float()
```

Qualifiers
```
static  // persists between calls
volatile   // usc  RAM (nice for ISR)
const  // make read-only
PROGMEM // use flash
```

Digital I/O
```
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int digitalRead(pin)
//Write High to inputs to use pull-up res
```

Analog I/O
```
analogReference([DEFAULT,INTERNAL,EXTERNAL])
int analogRead(pin) //Call twice if
switching pins from high Z source.
analogWrite(pin, value) // PWM
```

Advanced I/O
```
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
[MSBFIRST,LSBFIRST], value)
unsigned long pulseIn(pin, [HIGH,LOW])
```

Time
```
unsigned long millis() // 50 days overflow.
unsigned long micros() // 70 min overflow
delay(ms)
delayMicroseconds(us)
```

Math
```
min(x, y)    max(x, y)    abs(x)
constrain(x, minval, maxval )
map(val, fromL, fromH, toL, toH)
pow(base, exponent)  sqrt(x)
sin(rad)     cos(rad)     tan(rad)
```

Random Numbers
```
randomSeed(seed) // Long or int
long random(max)
long random(min, max)
```

Bits and Bytes
```
lowByte()        highByte()
bitRead(x,bitn)  bitWrite(x,bitn,bit)
bitSet(x,bitn)   bitClear(x,bitn)
bit(bitn) //bitn:  0-LSB 7-MSB
```

External Interrupts
```
attachInterrupt(interrupt, function,
[LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries:

Serial.
```
begin([300, 1200, 2400, 4800, 9600,
14400, 19200, 28800, 38400, 57600,
115200])
end()
int available()
int read()
flush()
print()
println()
write()
```

EEPROM (#include <EEPROM.h>)
```
byte read(intAddr)
     write(intAddr,myByte)
```

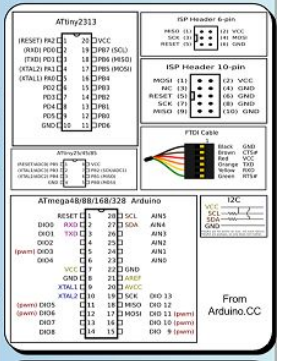
Servo (#include <Servo.h>)
```
attach(pin , [min_uS, max_uS])
write(angle) // 0-180
writeMicroseconds(uS) //1000-2000,
 1500 is midpoint
read() // 0-180
attached() //Returns boolean
detach()
```

SoftwareSerial(RxPin,TxPin)
```
// #include<SoftwareSerial.h>
begin(longSpeed) // up to 9600
char read() // blocks till data
print(myData) or println(myData)
```

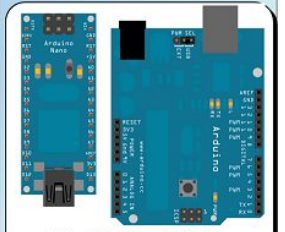
Wire (#include <Wire.h>) // For **I2C**
```
begin()          // Join as master
begin(addr) // Join as slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(mybyte)          // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission()     // Step 3
byte available() // Num of bytes
byte receive() //Return next byte
onReceive(handler)
onRequest(handler)
```

	ATMega168	ATMega328	ATmega1280
Flash (2k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (None has 14+8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 3 - (Int 1)	2,3,21,20,19,18 (IRQ0- IRQ5)
PWM pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0-13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



From Arduino.CC



Pics from Fritzing.Org under C.C. license

