Introduction To Robotics And Mechatronics

GianAndrea Müller

April 15, 2019

Contents

1 Basics **2**
1.1 Mechatronics Development . . 2
1.1.1 Sequential Approach . . 2
1.1.2 Concurrent Approach . 2
1.2 Robot Types 2

2 Labs **2**
2.1 00 2
2.1.1 Binary number formats 3
2.1.2 Conversion from Decimal to other systems . . 3
2.1.3 Hex Nibbles 3
2.1.4 Two's Complement . . . 3
2.1.5 Declaring numbers in C 3
2.1.6 Bitwise Operators . . . 4
2.2 05 - Lessons in Debugging . . . 4

3 Linux Commands and C Basics **4**
3.1 Basic File Manipulations 4
3.2 File naming 4
3.3 Wildcards 4
3.4 Escaping Conventions 4
3.5 Compilation 5
3.5.1 Variable Definition . . . 5
3.5.2 Variable Declaration . . 5
3.6 Dependencies and Makefiles . . 5
3.6.1 Macros in make 6
3.7 Pointers and Structures 7
3.8 Arithmetic Conversion 7

4 Sensors **7**
4.1 Dynamic Characteristics 8
4.2 Sampling 8
4.2.1 Analog-to-digital Conversion 8
4.2.2 Digital-to-Analog Conversion 8
4.3 Signal Analysis 9

5 Signal Processing **9**
5.1 Signal Conditioning 9
5.2 Signal Amplification 9
5.2.1 Ideal Amplifier (Opamp) 9
5.3 Signal Filtering 10

6 Digital Filtering **10**
6.1 Moving Average 11

7 Actuators **11**
7.1 Hydraulics 11
7.2 Pneumatics 11
7.3 Specific Energy 11
7.4 Solenoid 11
7.5 Voice Coil 11
7.6 Material deformation actuation 11
7.7 Electric Motors 12
7.8 Gearheads 13

8 Computer Vision **13**
8.1 Sensor technologies 13
8.2 Color Cameras 13

9 Robot Kinematics **13**
9.1 Control 13

10 Modelling and Control of Dynamic Systems **14**
10.1 Standard test signals 14
10.2 DC Motor Transfer Function . 14
10.3 Mass Spring Damper System . 14

11 Appendix **15**

1 Basics

Definition 1. ***Mechatronics*** *is defined as the field of study involving the analysis, designs, synthesis, and selection of systems that combine electronic and mechanical components with modern controls and microprocessors.*

Definition 2. *A* ***Robot*** *is defined as an actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks. A robot includes the control system and interface of the control system.*

- *Perception*
- *Manipulation*
- *Robot-Human Interaction*
- *Cognition*

1.1 Mechatronics Development

Definition 3. ***Integration*** *must start at an early stage to make systems less expensive, more reliable and more flexible.*

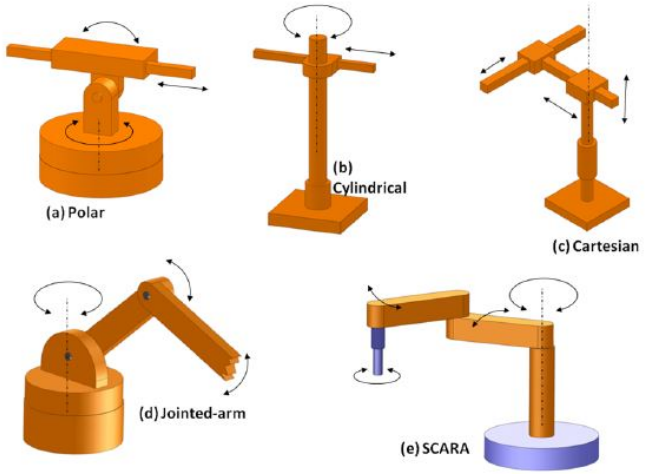
1.1.1 Sequential Approach

+ Well known design process
+ Centuries of experience with some systems
+ Designers able to focus on specific areas
- Bulky systems
- Complex mechanisms
- Non-adjustable movement cycles
- Constant speed drives
- Mechanical synchronization
- Rigid heavy structures
- Accuracy determined by tolerance of mechanism
- Manual control

1.1.2 Concurrent Approach

+ Compact
+ Simplified mechanisms
+ Programmable movements
+ Variable speed drives
+ Electronics synchronization
+ Lighter structures
+ Accuracy achieved by feedback
+ Automatic and programmable controls
- Difficult to coordinate
- More reliance on software control systems
- Broad experience needed by designers

1.2 Robot Types



2 Labs

2.1 00

A hexadecimal number is a number expressed by a base-16 numeral system which uses the symbols 0-9 and A-F. For conversion between hexadecimal and decimal apply the algorithms learned for binary numbers. On preference convert to binary first, using hex nibbles.

2.1.1 Binary number formats

Name	Size (bits)
Bit	1
Nibble	4
Byte	8
Word	16
Double Word	32

2.1.2 Conversion from Decimal to other systems

Dividing by the base Dividing the decimal number by the base of the target positional notation system and recording the resulting remainders until the integer part results in 0 delivers the digits of the number in reverse order.

Quotient	Integer Part	Remainder
18435/16	1152	3
1152/16	72	0
72/16	4	8
4/16	0	4

Subtraction Method Start with a weighted position greater than the number.

- If the number is greater than the weighted position for the digit, write down a 1 and subtract the weighted position value.
- If the number is less than the weighted position for the digit, write down 0 and subtract 0.

2.1.3 Hex Nibbles

Any of the hexadecimal digits can be understood as a nibble (4 bits). Therefore hexadecimal numbers can be viewed as a compact representation of binary numbers, since every hexadecimal bit directly translates to a certain nibble as listed below.

hex	bin	dec	hex	bin	dec
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	A	1010	10
3	0011	3	B	1011	11
4	0100	4	C	1100	12
5	0101	5	D	1101	13
6	0110	6	E	1110	14
7	0111	7	F	1111	15

2.1.4 Two's Complement

Writing negative numbers in the two's complement does not simply use the first bit in a binary number as a sign bit, but assigns that bit its negative positional value. Thereby the double-encoding of 0 (negative 0 respectively) is omitted.

Two's complement					
Bits	Unsigned value	2's	Bits	Unsigned value	2's
0000	0	0	1000	8	-8
0001	1	1	1001	9	-7
0010	2	2	1010	10	-6
0011	3	3	1011	11	-5
0100	4	4	1100	12	-4
0101	5	5	1101	13	-3
0110	6	6	1110	14	-2
0111	7	7	1111	15	-1

How to find the Two's Complement Representation

1. Find the binary representation of the magnitude of the number.
2. Find the binary complement of that representation. Flipping 0 to 1 and vice versa.
3. Add one to the complement.

The above method applied to hexadecimal numbers requires the conversion to binary and back. This can be circumvented using the following approach:

1. Subtract each hexadecimal digit from 15.
2. Add 1 to the resulting number.

2.1.5 Declaring numbers in C

```
1 #include <stdint.h>
2 uint16_t a = 0x4803;
```

Since the built in data types of C are ambiguous, depending on the system used on. For that reason `stdint.h` is used, making the following data types available:

Type declaration	Bits	Type	Range
int8_t	8-bit	integer	-128 - 127
uint8_t	8-bit	unsigned integer	0 - 255
int16_t	16-bit	integer	-32768 - 32767
uint16_t	16-bit	unsigned integer	0 - 65535
int32_t	32-bit	integer	-2'147'483'648 - 2'147'483'647
uint32_t	32-bit	unsigned integer	0 - 4'294'967'295
int64_t	64-bit	integer	$-9.22 \cdot 10^{18}$ - $9.22 \cdot 10^{18}$
uint64_t	64-bit	unsigned integer	0 - $1.84 \cdot 10^{19}$

2.1.6 Bitwise Operators

Bitwise operators work on bits and perform bit-by-bit operations. The truth tables for bitwise AND `&`, bitwise OR `|` and bitwise exclusive OR (XOR) `^` are:

p	q	p&q	p|q	p^q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

`&`	Binary AND Operator copies a bit to the result if it exists in both operands.
`|`	Binary OR Operator copies a bit if it exists in either operand.
`^`	Binary XOR Operator copies the bit if it is set in one operand but not both.
`~`	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.
`<<`	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.
`>>`	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

2.2 05 - Lessons in Debugging

1. Reboot.
2. Check all wire connections thoroughly.
3. Check if the hardware is functional.
4. Check if all variables are initialized.
5. Check if the array size is correct.
6. Check initial conditions for loops.
7. Check incrementation for loops.
8. Check for potential segmentation faults.
   - For example advancing over array in the wrong direction.
9. Check for integer divisions.
10. Check result reproducibility to detect random successes.
11. Output values, include unit tests.
12. Check superfluous semicolons.

3 Linux Commands and C Basics

3.1 Basic File Manipulations

Command	Function	MSDOS Equivalent
cp	Copy a file or collection of files	COPY
mv	Move or rename a file or collection of files	COPY REN
rm	Remove a file or collection of files	DEL
ln	Create a link (or alias) from one file to another	N/A
ls	List a directory. There are a lot of options	DIR
mkdir	Create a directory	MKDIR
rmdir	Remove a directory (must be empty)	RMDIR

3.2 File naming

- Maximum name length: 255, the path can be longer,.
- All characters besides `'/'` and NUL are allowed.
- File names are case sensitive.
- File names do not have components or extensions. Many applications however regard the part of a file name after the last dot as having special significance.

3.3 Wildcards

`*`	Matches any sequence of characters.
`?`	Matches any single character.
`[..]`	Matches any of the characters included in the list between the brackets. Ranges may also be specified using a dash so, for example `[a-z]` means any lower case letter.
`[!..]`	Matches any character not included in the list.

3.4 Escaping Conventions

*	File naming wild card	?	File naming wild card
[	File naming wild card	]	File naming wild card
$	Shell variable substitution	&	Shell backgrounding
<	Input rederection	>	Output rederection
|	Shell piping	;	Shell command grouping
(	Shell command grouping	)	Shell command grouping
{	Shell variable substitution	}	Shell variable substitution
\	Shell escaping	"	Shell escaping
‘	Shell escaping	’	Shell command substitution

In addition ^ and ! can cause problems when used in file names. Files beginning with - are also problematic.
It is possible to overcome many of the constraints by escaping file names containing awkward characters using quotes to enclose them or preceding awkward characters with a backslash.

3.5 Compilation

`gcc anyname.c`	Compile a program in c with any name desired.
`cc myprog.c`	Compile a program in c called myprog.
`cc myprog.c 2>myprog.errs`	Capturing compilation errors.
`cc -o myprog myprog.c`	Specific output file name.
`cc -o myprog myprog.c -lm -ls626`	Inclusion of math and sensoray libraries.
`cc -c green.c`	Stop compilation after the assembler stage, producing `green.o`.

Separate compilation steps

1. Compile green.o: `cc -c green.c`
2. Compile blue.o: `cc -c blue.c`
3. Link the parts together: `cc green.o blue.o`

Splitting a program

- Be sure no two files share function names.
- Be sure no two files share global variable names.
- Declare global variables in the header file: `extern int globalvar`.
- To use functions from another file make a .h file with the function prototypes and use `#include` to include those .h files within your c files.
- At least one of the files must have a main() function.

3.5.1 Variable Definition

A variable definition tells the compiler where and how much storage to create for the variable. A variable definition specifies a data type and name. Multiple names can be introduced at once if separated by commas.

```
1 int i,j,k;
```

This line both declares and defines the variables i,j and k. Direct initialization is also possible:

```
1 int d = 3, f = 2;
```

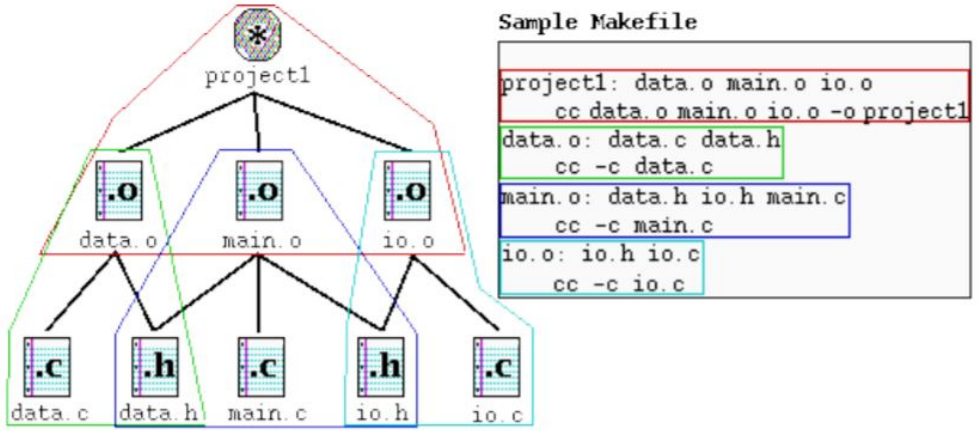
If a variable is left uninitialized its value is undefined in general. However, variables with static storage duration are implicitly set to 0.

3.5.2 Variable Declaration

It is possible to declare a variable without defining it. The declaration is accepted during compilation but has to be fitted with a definition at the time of linking of the program. This means that if a program consists of multiple files you can declare your variable wherever you need it but only define it once, since multiple definitions of the same variable are prohibited.

```
1 //Variable declaration (everywhere needed)
2 extern int a,b; //Compiler knows that the variable exists
3
4 //Variable definition (only once)
5 int a,b; //Compiler knows that the variable exists
6          //And allocates the storage space needed
```

3.6 Dependencies and Makefiles



A makefile encodes the dependencies of different parts of the program, enabling building the code in an optimized manner by only compiling the part of the dependency tree that was affected by a change. Note that only files listed in the makefile will lead to an update when changed. In the example above changing the .h-files for example will not do that.

```
make //will execute 'Makefile' or 'makefile'
make -f otherfile //accepts any filename
```

3.6.1 Macros in make

```
OBJECTS = data.o main.o io.o
project1: $(OBJECTS)
  cc $(OBJECTS) -o project1
data.o: data.c data.h
  cc -c data.c
main.o: data.h io.h main.c
  cc -c main.c
io.o: io.h io.c
  cc -c io.c
```

A macro can also be specified when running make. This overrides the value of OBJECTS within the above specified makefile.

```
make 'OBJECTS=data.o newio.o main.o' project1
```

Special macros

- `CC` Contains the current C compiler. Defaults to cc.
- `CFLAGS` Special options which are added to built-in C rule.
- `$@` Full name of the current target.
- `$?` A list of files for current dependency which are out-of-date.
- `$<$` The source file of the current (single) dependency.

Manipulate macro evaluation Assuming `OBJS=data.o io.o main.o` the macro can be reinterpreted using

```
$(OBJS:.o=.c) //data.c io.c main.c
```

Predefined rules make knows that to create an .o file the corresponding .c file is needed. Thus these .c files can be omitted. Why are the corresponding make commands not repeated?

```
OBJECTS = data.o main.o io.o
project1: $(OBJECTS)
  cc $(OBJECTS) -o project1
data.o: data.h
main.o: data.h io.h
io.o: io.h
```

When compiling programs on Wiliki add on top:

```
CFLAGS=-Aa -D_HPUX_SOURCE
```

which allows make to use ANSI standard C compilation.

Shortcuts There can be more than one file in the target section of the dependency rules. If a file appears in several target sections all its sources are gathered.

```
CFLAGS = -Aa -D_HPUX_SOURCE
OBJECTS = data.o main.o io.o
project1: $(OBJECTS)
  cc $(OBJECTS) -o project1
data.o main.o: data.h
io.o main.o: io.h
```

3.7 Pointers and Structures

```c
struct tag {
  char lname[20]; /* last name */
  char fname[20]; /* first name */
  int age; /* age */
  float rate; /* e.g. 12.75 per hour */
};
```

The following file declares an instance of the structure, assigns data and then prints it out.

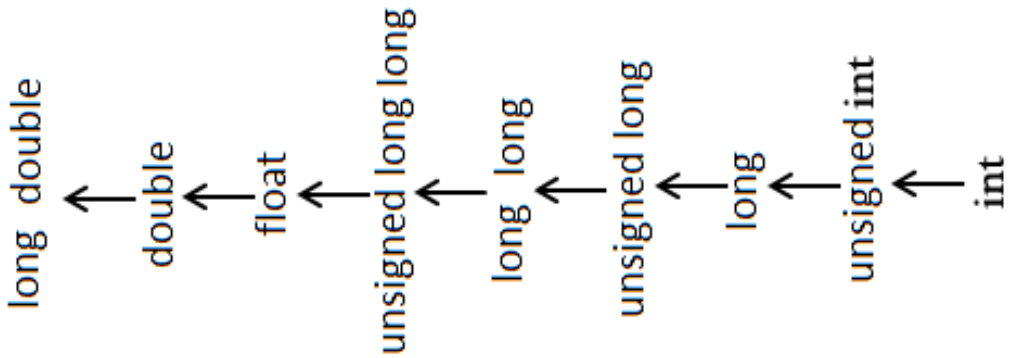
```c
struct tag my_struct; /* declare the structure my_struct
   */
int main(void)
{
strcpy(my_struct.lname,"Jensen");
strcpy(my_struct.fname,"Ted");
printf("\n%s ",my_struct.fname);
printf("%s\n",my_struct.lname);
return 0;
}
```

- In original C we cannot pass whole structures to functions, in ANSI standard C however that is possible.

```c
struct tag *st_ptr; //declare a pointer to type <struct
   tag>
st_ptr = &my_struct; //let it point to the above instance
(*st_ptr).age = 63; //dereference it for access
st_ptr->age = 63; //dereferencing and access in one
```

3.8 Arithmetic Conversion

Whenever an arithmetic operator encounters variables of different types they are cast to a common type following the hierarchy below:



4 Sensors

Definition 4. ***Range*** *The* ***range*** *of a sensor defines the limits between which the input can vary.*

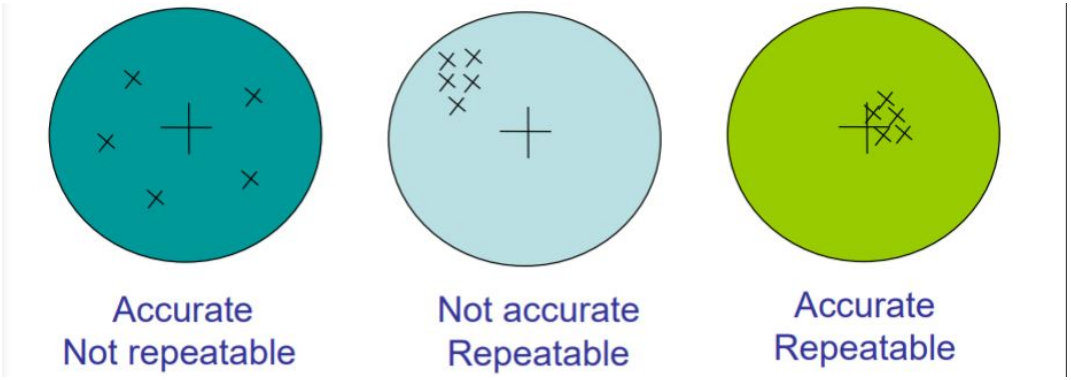
Definition 5. *The* ***span*** *of a sensor is the maximum value of the input minus the minimum value.*

Definition 6. *The* ***sensitivity*** *of a sensor is a relationship that indicates how much output per unit input the sensor produces.*

Definition 7. *The* ***accuracy*** *of a sensor captures the extent to which the values indicated by a measurement system might be wrong.*

Definition 8. *The* ***repeatability*** *of a sensor describes its ability to give the same output for repeated applications of the same input value.*

$$repeatability = \frac{value_{max} - value_{min}}{range} \cdot 100$$



Definition 9. *The **resolution** of a sensor is the smallest change in the input value that will produce an observable change in the output.*

Definition 10. *The **stability** of a sensor is its ability to give the same output when used to measure a constant input over a period of time.*
***Drift** is often used to describe the change of output over time.*
***Zero drift** is used of changes that occur in output when there is zero input.*

Definition 11. *The **error** describes the difference between the result of the measurement and the true value of the quantity being measured.*

Definition 12. *The **hysteresis error** describes the difference between two measurements of the same value once measured on an increasing and once measured on a decreasing change.*

Definition 13. *The **non-linearity error** describes how much error arises from the assumption that the transducer has a linear relationship between input and output.*

Definition 14. *The **static characteristics** of a sensor describe its behaviour in the steady state.*

4.1 Dynamic Characteristics

Definition 15. *The **dynamic characteristics** of a sensor describe its transient behaviour before reaching the steady state.*

Definition 16. *The **response time** of a dynamic system is the time that elapses after a constant input is applied until a certain percentage of the input is reached.*

Definition 17. *The **time constant** of a dynamic system is a measure of the inertia. It describes how fast the system reacts. The larger the time constant, the slower the reaction.*

Definition 18. *The **rise time** of a dynamic system is the time that elapses after until the output rises to some specified percentage of the steady-state output.*

Definition 19. *The **settling time** is the time taken for the output to settle to within some percentage of the steady-state value.*

4.2 Sampling

$\boxed{f_s > 2f_{\max}}$ Nyquist's Sampling Theorem

f_s	Sampling rate	Hz
$2f_{\max}$	Nyquist frequency	Hz

4.2.1 Analog-to-digital Conversion

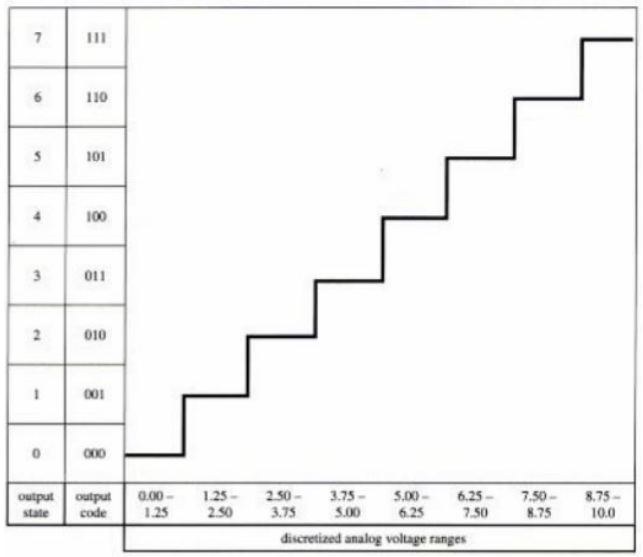
1. Quantization: Transformation of a continuous analog signal into a set of discrete output states.
2. Coding: Assignment of digital codes to each output state.

Definition 20. ***Resolution** of an ADC is the number of bits used to approximate the analog value of the input.*

$$\boxed{N = 2^n}$$

N	*Number of possible states*
n	*Number of bits*
$N-1$	*Number of decision points*

*The **maximum error** is $\pm 1/2$ LSB (**Least Significant Bit**).*



Definition 21. *The **aperture time** is the time window during which the output has any error due to changes of the input.*

Definition 22. ***Dithering** is the process of adding random noise to a signal before discretizing it order to randomize the quantization error.*

4.2.2 Digital-to-Analog Conversion

- Inmpulse train
  - Difficult to do exactly in electronics
  - Does not preserve original spectrum
- Sample and hold
  - Equivalent to sample and hold from ADC

Definition 23. *A **reconstruction filter** is used to create a smooth analog signal from the digital output of a DAC.*

4.3 Signal Analysis

Definition 24. *The **mean** is the average of a signal an correspond to the DC current of a signal.*

$$\mu = \mathbb{E}[X] = \frac{1}{N}\sum_i x_i = \sum_{-\infty}^{\infty} xPDF(x)dx$$

Definition 25. *The **standard deviation** σ describes how far the i^{th} sample deviates from the mean.*

$$\begin{aligned}\sigma^2 &= \mathbb{E}\left[(X-\mu)^2\right] \\ &= \frac{1}{N-1}\sum_i (x_i-\mu)^2 \\ &= \int_{-\infty}^{\infty}(x-\mu)^2 PDF(x)dx\end{aligned}$$

σ^2 *is called the **variance** of the signal.*

Definition 26. *The **signal to noise ratio (SNR)** is defined as*

$$SNR = \frac{\mu}{\sigma}$$

Definition 27. *The **coefficient of variation (CV)** is defined as*

$$CV = \frac{\sigma}{\mu}100\%$$

$\boxed{P(x) = \frac{1}{\sqrt{2\pi\sigma}}e^{-(x-\mu)^2/2\sigma^2}}$ Normal Distribution

5 Signal Processing

Definition 28. ***Noise** can be described as an undesired disturbance in a measurement signal.*

Noise can be caused by

- Mechanical Vibrations and Acoustic Sources
- Thermal effects
- Electrical effects
  - **Shot (Schottky) Noise** arises due to non-continuous or quantized nature of electron transport. Stops when current flow stops, has flat PSD, is independent of temperature.

$$I_n = \sqrt{2qI_{dc}\Delta\omega}$$

  - **Johnson-Nyquist Noise** is caused by thermal agitation of electrons at thermal equilibrium.

$$V_n = \sqrt{4k_BTR} \text{ in } \mathrm{V}/\sqrt{\mathrm{Hz}}$$

Voltage Variation per root hertz bandwith

- Quantization effects (maximum error of $\pm\frac{1}{2}$ LSB)
- Numerical effects (fixed number of bits, representation errors)
  - Additive error increases with N
  - Random error increases with $\sqrt{N}$

5.1 Signal Conditioning

Reasons for conditioning:

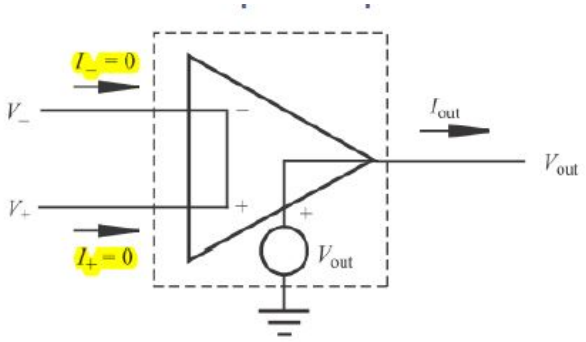
- Correct magnitude and phase
- Amplification
- Interference Removal
- ADC, DAC
- Safety

Examples:

- Fuses to prevent damage to the next element
- Circuit isolations (opto-couplers) delivering information encoded as light
- Polarity protection
- Voltage limitation

5.2 Signal Amplification

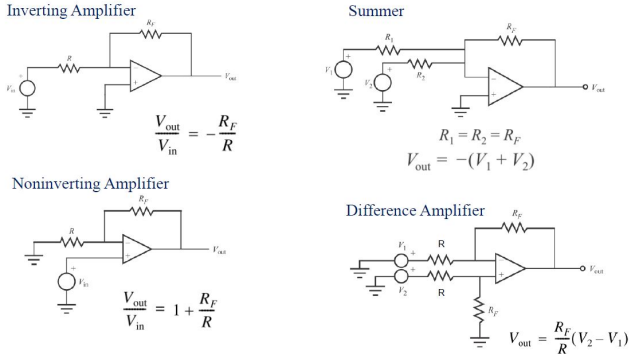
5.2.1 Ideal Amplifier (Opamp)



- Infinite impedance at inputs $I_+ = I_- = 0$
- Infinite gain $V_0 = K_{\to\infty}(V_+ - V_-)$

- Zero input impedance
- Closed loop gain: $K_{closed} = \frac{K_{open}}{1+K_{open} \cdot Feedback}$

  where $Feedback$ denotes how much of the output signal (in percent) is fed back.
- Closed loop gain for large open loop gains: $K_{closed} = \frac{1}{Feedback}$



5.3 Signal Filtering

Definition 29. ***Filtering*** *is used to describe the process of removing a certain band of frequencies from a signal and permitting others to be transmitted.*

Definition 30. *A* ***passive filter*** *is built only from resistors, capacitors and inductors. An* ***active filter*** *also includes operational amplifiers.*

Definition 31. ***Digital filtering*** *describes the process of filtering the signal after it has been sampled.*

Check out the lecture for details on Fouriertransform, Laplacetransfrom, Time vs. Frequency domain and the implementation of a range of basic filters.

6 Digital Filtering

Definition 32. ***Filters*** *are used for signal restoration (removal of unwanted parts of the signal such as noise) and signal separation (extraction of useful parts of the signal).*

- Analog filters
  - \+ Cheap
  - \+ Fast
  - \+ Have a large dynamic range in amplitude and frequency
- Digital filters
  - \+ Programmable / Versatile
  - \+ Can handle slow signals
  - \+ No drift / temperature dependency
  - \- Introduce a delay into the signal (more components in the signal path)
  - \- Relatively expensive
  - \- Not impervious to noise

Filtering approaches

- **Convolution**: Weighted average of samples of the input.
- **Recursion** Weighted average of samples of the input and previously calculated samples of the output.

Filter characteristics

Definition 33. ***FIR*** *filters have a finite impulse response, thus the response settles to zero in finite time.*

- \+ *Require no feedback, thus rounding errors are not accumulated and the computation is simpler.*
- \+ *Inherently stable.*
- \+ *When having a symmetric coefficient sequence they automatically have linear phase.*
- \- *Need more computation power, especially for low frequency cutoffs.*

Definition 34. ***IIR*** *filters have a frequency response that continues infinitely. Analog filters (resistance, capacitors / inductors) are generally IIR filters.*

- \+ *Efficiently implemented for certain requirements (passband, stopband, ripple, roll-off). The same requirements can be met with a lower order IIR than FIR.*
- \+ *Efficient calculation*
- \- *Harder to design*
- \- *Nonlinear phase likely, linear phase only approximately achievable*

Definition 35. *A filter with a* ***linear phase*** *has a linear dependency between frequency and phase response.*

Definition 36. *A filter with a nonlinear phase can have* ***phase distortion****, which leads to a change in the shape of the waveform.*

Definition 37. ***Stopband attenuation*** *describes how fast the filters response decays for frequency above the passband.*

- \+ Fast step response

+ No overshoot

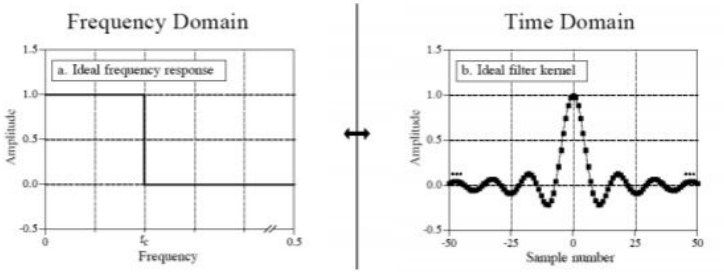
+ Linear phase (no phase distortion)

+ Fast roll-off

+ No ripple in passband

+ Good stopband attenuation

6.1 Moving Average



The inverse fourier transform of the ideal frequency response produces a sinc function.

$$h[i] = \frac{\sin(2\pi f_c i)}{i\pi}$$

This is easily calculated but not so easily implemented on a computer since it is infinite in time. The solution is to apply a smooth window to the time domain signal in order to avoid an abrupt ending of the IR of the filter. For that purpose Blackman or Hann Windows are applied.

7 Actuators

7.1 Hydraulics

+ High power/weight ratio

+ Low time constants

+ High stiffness due to fluid incompressibility

- High cost due to the hydraulic system

- Space requirements lead to difficult miniaturization

- Risk of oil leaks (messy)

7.2 Pneumatics

+ Availability and cleanliness of air

+ Leaks are not dangerous

- Lower energy density than hydraulics

- Compressibility of air leads to imprecise kinematic control

7.3 Specific Energy

Definition 38. *The **specific energy** of an actuator refers to the amount of energy it can produce in a given volume.*

For pneumatic/hydraulic actuators the specific energy corresponds to the pressure: $w_{hdy} = w_{pne} = p$, for electrochemical actuators the specific energy is: $w_e = \frac{1}{2}B^2/\mu_0$.

Type	Limiting factor	Specific energy
Hydraulic	$p_{max} = 70\,\text{bar}$	$7 \times 10^6\,\text{J m}^{-3}$
Pneumatic	$p_{max} = 6\,\text{bar}$	$0.6 \times 10^6\,\text{J m}^{-3}$
Electric	$B_s = 1\,\text{T}$	$0.4 \times 10^6\,\text{J m}^{-3}$

where B_s corresponds to the saturated magnetic field.

7.4 Solenoid

Definition 39. *A **solenoid** consists of a coil and a movable iron core called **armature**. A solenoid can act as a relay when establishing an electrical contact.*

7.5 Voice Coil

Definition 40. *A **voice coil** consist of a coil that moves in a magnetic field produced by a permanent magnet and intensified by an iron core.*

7.6 Material deformation actuation

+ High energy density

+ High rigidity

+ High resolution

- Small deformations (mechanical amplification necessary)

Possible approaches

- Thermal deformation through resistive heating
- Conducting polymers
  + High force
  + High deformations
  - Low frequencies

- Magnetostriction
- Shape memory alloys (switching between different crystalline structures through heating
- Piezo drives
  + Can be used to generate displacement/force or a combination of both

$$F_{max} \approx k_T \Delta L_0$$

$$F_{max,eff} \approx k_T \Delta L_0 \left(1 - \frac{k_T}{k_T + k_S}\right)$$

ΔL_0	maximal nominal displacement
k_T	piezo actuator stiffness
k_S	external spring stiffness (environment)

- Slip-Stick Actuator

  A piezo actuator is attached to a mobile base. It slowly extends while the base remains stationary and retracts quickly such that the base slips.

$$v = (L - \delta) f$$

v	velocity	f	frequency
L	full elongation		
δ	backlash		

7.7 Electric Motors

- Brushless (vs. Brushed)
  + High power density
  + Easier manufacture
  + Less wear/tear, low heat generation
  + Quicker speed adjustment with electronic control
- Induction Motor
  + Simple and low-cost construction
  + Low maintenance costs
  + Can be used in hazardous environments, since it produces no sparks
- Servomotor
  + Precise control of displacement, velocity and acceleration
  + Incorporates closed-loop servomechanism
- Stepper
  + Simple construction
  + Open loop control (no position sensor necessary
  + High starting torque
  + Can rotate in both directions
  + Can move in precise angular increments
  - Slip if overloaded
  - Preferential stopping positions
  - Vibrations
  - Low maximal speed

$$V_{emf} = k_e \omega$$

Back electromotive force (EMF)

$$\boxed{V_{in} = \underbrace{L\frac{dI_{in}}{dt}}_{\text{neglectable}} + RI_{in} + k_e \omega}$$

$$\boxed{T = k_t I_{in}}$$

k_e	electrical motor constant
k_t	torque motor constant

$$\boxed{T = \left(\frac{k_t}{R}\right) V_{in} - \left(\frac{k_e k_t}{R}\right) \omega}$$

$$T(\omega) = T_s \left(1 - \frac{\omega}{\omega_{max}}\right)$$

T_s	Stall torque ($\omega = 0$)
ω_{max}	No-load speed ($T = 0$)

$$T_s = \left(\frac{k_t}{R}\right) V_{in} \qquad \omega_{max} \frac{T_s R}{k_e k_t}$$

$$\boxed{P(\omega) = T\omega = \omega T_s \left(1 - \frac{\omega}{\omega_{max}}\right)}$$

The maximum power is obtained at $\omega^* = \frac{1}{2}\omega_{max}$.

7.8 Gearheads

- Spur gearhead
  - \+ Low friction per train
  - \+ Input and output shaft need not be aligned
  - \- Low torque
- Planetary gearhead
  - \+ Higher reduction ratio per train and higher friction
  - \+ Higher torques
  - \+ High performance in small space

$$\omega_{load} = \frac{1}{r}\omega_{motor} \qquad T_{load} = rT_{motor} \qquad J^{equ}_{motor} = \frac{1}{r^2}J_{load}$$

r gear ratio
J inertia

The optimal gear ratio for load acceleration and for minimizing the dissipated energy is:

$$r_{opt} = \sqrt{\frac{J_{load}}{J_{motor}}}$$

$$P_{input} = V_{in} \cdot I_{in}$$
$$P_{output} = T\omega$$
$$P_{loss} = T_{friction}\omega + R_a I^2 \eta \qquad = \frac{P_{output}}{P_{input}}$$

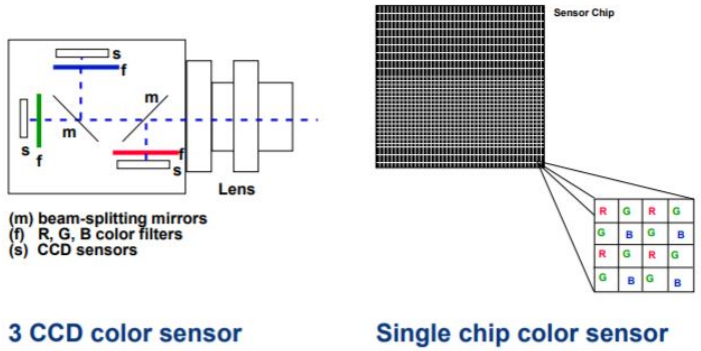
8 Computer Vision

Definition 41. *A **camera** is a device comprising optics, an imaging sensor, digitization hardware and optionally computer vision algorithms reproducing a view of the world.*

8.1 Sensor technologies

- **CCD** (Charged Coupled Device)
  - \+ High sensitivity
  - \+ High dynamic range
  - \- Expensive
  - \- Needs off-chip circuitry
- **CMOS** (Complementary Metal-Oxide Semiconductor)
  - \+ All electronics on a single chip
  - \+ Much cheaper

8.2 Color Cameras



9 Robot Kinematics

Definition 42. *The **Degrees of Freedom (DOF)** of a robotic system defines the number of independent movements the system can perform with respect to a coordinate system.*

Definition 43. *The **forward kinematics** of a robotic system describe the position and orientation of the end effector depending on the joint angles.*

Definition 44. *The **inverse kinematics** of a robotic system describe the joint angles depending on a desired end effector position.*

Definition 45. *A **redundant robot** has more joints than the DOF of the task space.*

$$\begin{bmatrix} \dot{x}_{1,ee} \\ \vdots \\ \dot{x}_{n,ee} \end{bmatrix} = \underline{J} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix}$$

$\underline{\dot{x}}_e e$ velocity of the end effector
$\underline{\dot{q}}$ joint velocities
$\underline{J}$ Jacobian

Definition 46. *A **path** is a locus of points in the task space is a purely geometric description of motion.*
*A **trajectory** is a path on which a time law is defined. It is a spatio-temporal description of motion.*

9.1 Control

A simple approach to controlling a multi-degree of freedom robot is controlling each joint separately.

10 Modelling and Control of Dynamic Systems

Laplace Transform:

$$\boxed{\begin{array}{c} L_t\left[f(t)\right](s) \equiv \int_0^\infty f(t)e^{-st}dt \\ f(t) \equiv \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} F(x)e^{st}ds \end{array}}$$

Properties

- Linearity
- First derivative

$$L_t\left[\tfrac{d}{dt}f(t)\right] = sF(x) - f(0)$$

- Second derivative

$$L_t\left[\tfrac{d^2}{dt^2}f(t)\right] = s^2F(s) - sf(0) - \tfrac{d}{dt}f(0)$$

- Integral

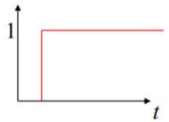
$$L_t\left[\int f(t)dt\right] = \tfrac{1}{s}F(s)$$

10.1 Standard test signals

- Step Input

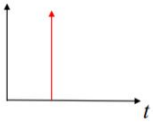
$x = 0 \quad t = t_0$

$x = 1 \quad t > t_0$



- Unit Impulse

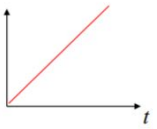
$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$



- Ramp Input

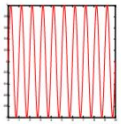
$x = 0 \quad t = t_0$

$x = t \quad t > t_0$



- Sinusoidal Input

$x = A\sin(\omega t)$



10.2 DC Motor Transfer Function

$$\boxed{\begin{array}{c} V_{armature} = L\frac{dI_{armature}}{dt} + RI_{armature} + K_{backemf}\omega, \quad (K_{backempf}\omega = e_b) \\ V(s) = (sL+R)I(s) + K_b s\Theta \\ I(s) = \frac{V(s) - sK_b\Theta}{Ls+R} \\ \Theta(s) = \frac{K_t I(s)}{s(Js+b)} \end{array}}$$

neglecting the armature inductance:

$$\frac{\Theta(s)}{V(s)} = \frac{K_t}{s^2RJ + s(Rb + K_tK_b)}$$

where we define $K_m = K_t(Rb + K_tK_b)$ and $T_m = RJ/(Rb + K_tK_b)$:

$$\boxed{\begin{array}{c} \frac{\Theta(s)}{V(s)} = \frac{K_m}{s(T_m s+1)} \\ \frac{\Omega(s)}{V(s)} = \frac{s\Theta(s)}{V(s)} \end{array}}$$

and thus

$$\boxed{K_m(1 - e^{-t/T_m}),\ t \geq 0}$$

10.3 Mass Spring Damper System

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2+cs+k} = G(s)$$

$$G(s) = \frac{1}{k}\frac{(k/M)}{s^2+(\frac{c}{M})s+(k/M)} \qquad \omega_n = \sqrt{k/M}, \zeta = \frac{c}{2\sqrt{kM}}$$

Step answer, underdamped case:

$$x(t)\left[1 - e^{-\zeta\omega_n t}\left(\cos(\omega_d t) + \frac{\zeta}{\sqrt{1-\zeta^2}}\sin(\omega_d t)\right)\right]$$

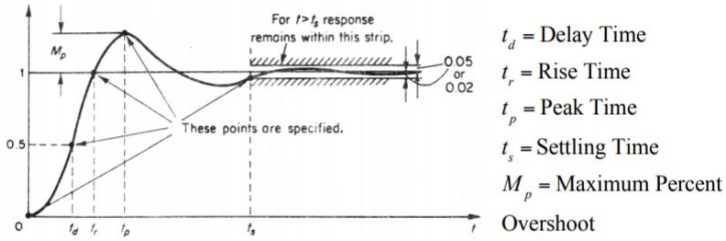
$$\omega_d = \omega_n\sqrt{1-\zeta^2}$$

Step answer, critically damped case:

$$x(t) = \tfrac{1}{k}\left[1 - e^{-\omega_n t}\left(1 + \omega_n t\right)\right]$$

Step answer, overdamped case:

$$x(t)\left[\frac{\omega_n}{2\sqrt{\zeta^2-1}}\left(\frac{e^{P_1 t}}{P_1} - \frac{e^{-P_2 t}}{P_2}\right)\right] \qquad P_1\zeta + \sqrt{\zeta^2-1},\ P_2 = \zeta - \sqrt{\zeta^2-1}$$



11 Appendix

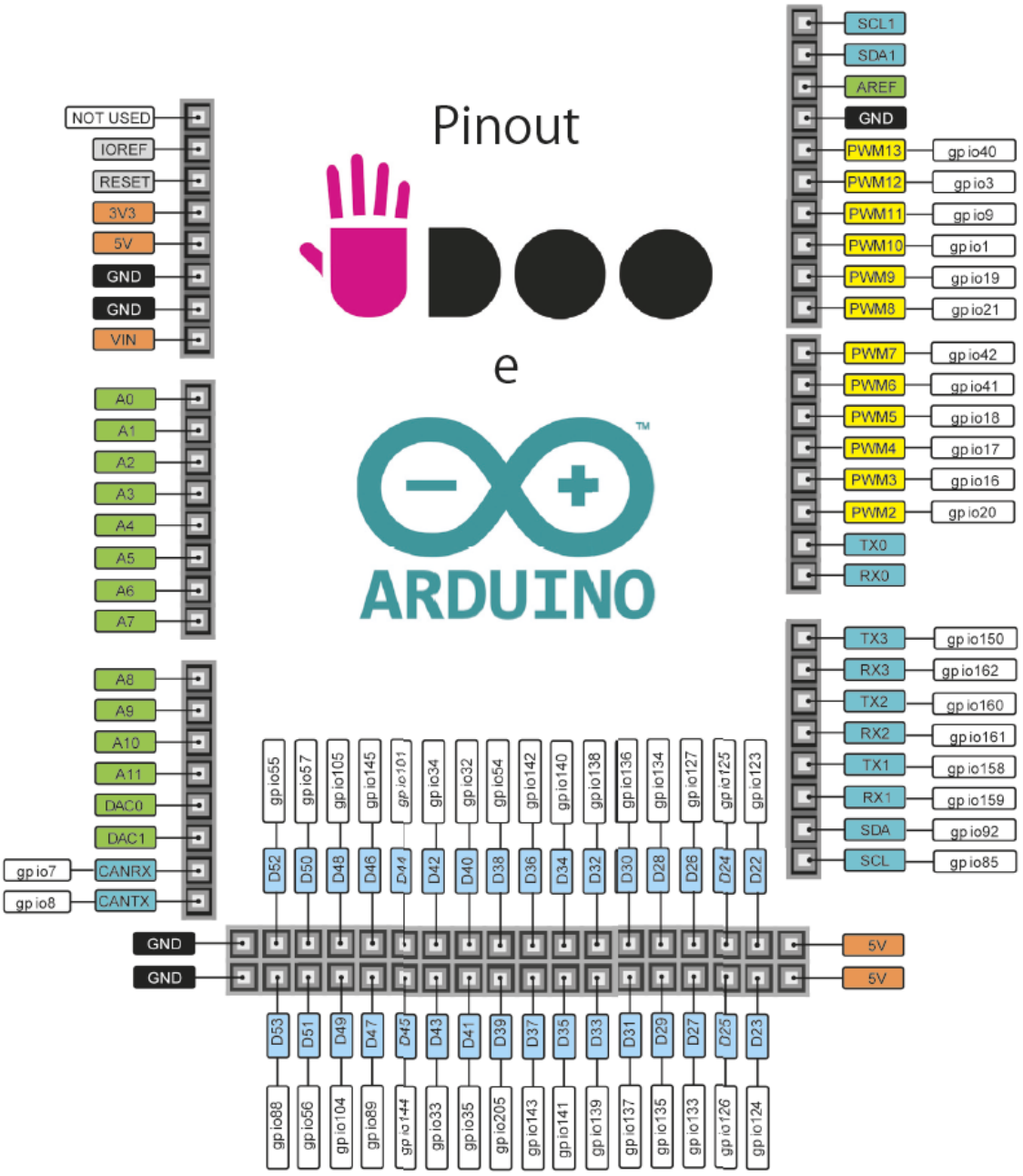


Figure C.1: UDOO pinout

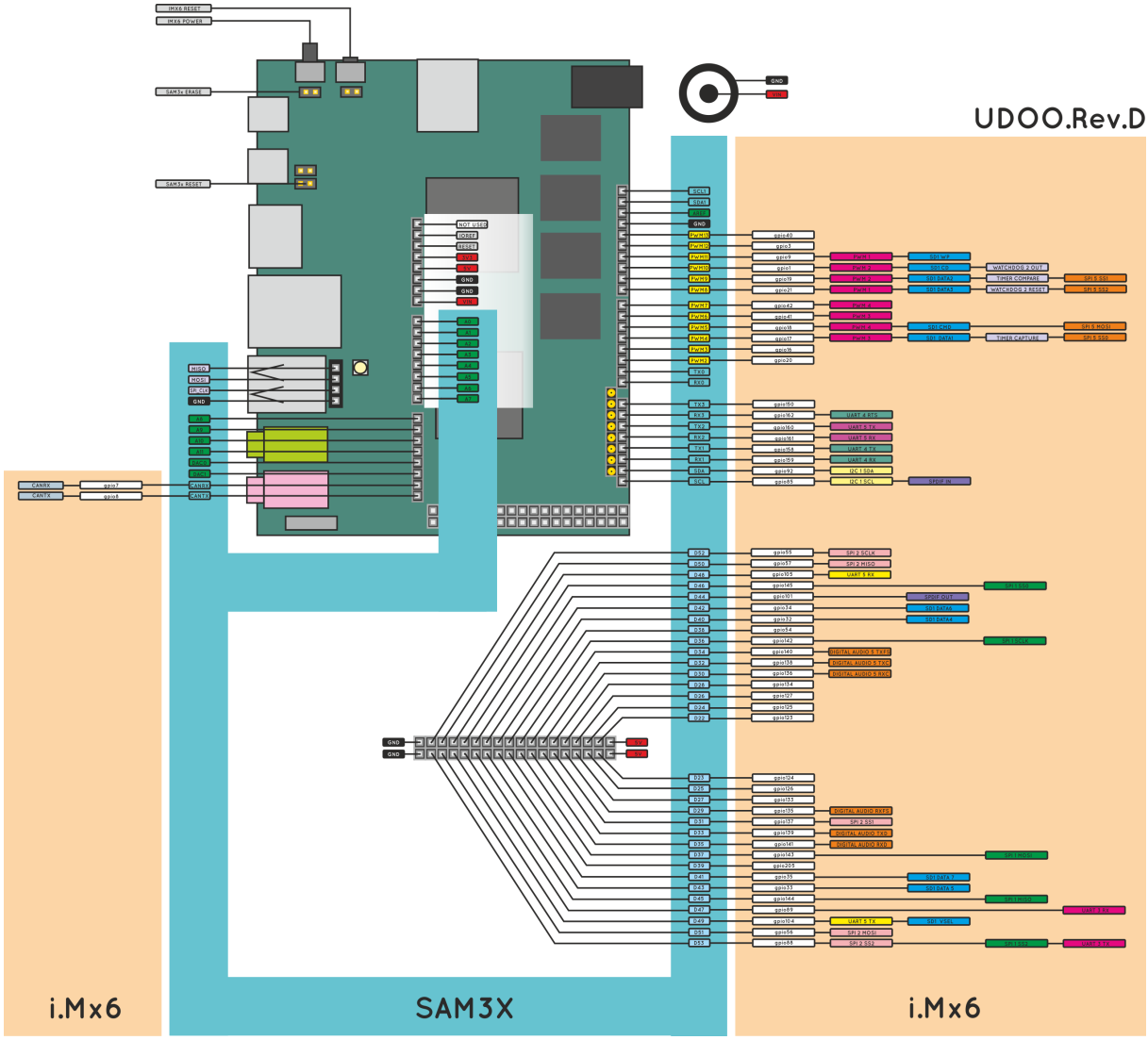


Figure C.2: UDOO pinout diagram.

IMX6 - SAM3X PHYSICAL PINS

This table describes UDOO pinout functions. Some of these pins are shared between the two processors. Their function can be choosed via software muxing.
Every i.Mx6 and SAM3X pin can carry an alternate function. Up to 8 alternate function are available for i.Mx6. The table below gives a quick over view.

	UDOO Pin Connector		SAM3X						i.MX6										
	Bank	Physical pin	Physical pin	Pin Name	Arduino function	Possible functions	GPIO function	Physical pin	Pin Name	Pad Mode ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	GPIO function
IOL	J17	1	27	URXD/PA8	RX0	URXD / PWMH0 / WKUP4	IN/OUT	M3	CSI0_DAT11	IPU1_CSI0_DATA11	AUD3_RXFS	ECSPI2_SS0	UART1_RX_DATA		GPIO5_IO29		ARM_TRACE08	IN/OUT
	J17	2	2	UTXD/PA9	TX0	UTXD / PWMH3	IN/OUT	M1	CSI0_DAT10	IPU1_CSI0_DATA10	AUD3_RXC	ECSPI2_MISO	UART1_TX_DATA		GPIO5_IO28		ARM_TRACE07	IN/OUT
	J17	3	144	PB25	PWM2	RTS0 / TIOA0	IN/OUT	D20	SD1_CLK	SD1_CLK	ECSPI5_SCLK	OSC32K	GPT_CLKIN		GPIO1_IO20			IN/OUT
	J17	4	139	PC28	PWM3	A7 / TIOA7	IN/OUT	A21	SD1_DAT0	SD1_DATA0	ECSPI5_MISO	CAAM_WRAPPER	GPT_CAPTURE1	PCIE_CTRL	GPIO1_IO16			IN/OUT
	J17	5	112 -137	PA29-PC26	PWM4	SPI0_NPCS1 / NRD A5 / TIOB6	IN/OUT	C20	SD1_DAT1	SD1_DATA1	ECSPI5_SS0	PWM3_OUT	GPT_CAPTURE2		GPIO1_IO17			IN/OUT
	J17	6	136	PC25	PWM5	A4 / TIOA6	IN/OUT	B21	SD1_CMD	SD1_CMD	ECSPI5_MOSI	PWM4_OUT	GPT_COMPARE1		GPIO1_IO18			IN/OUT
	J17	7	135	PC24	PWM6	A3 / PWML7	IN/OUT	B19	SD4_DAT1	SD4_DATA1	PWM3_OUT				GPIO2_IO09			IN/OUT
	J17	8	134	PC23	PWM7	A2 / PWML6	IN/OUT	F17	SD4_DAT2	SD4_DATA2	PWM4_OUT				GPIO2_IO10			IN/OUT
	J21	1	133	PC22	PWM8	A1 / PWML5	IN/OUT	F18	SD1_DAT3	SD1_DATA3	ECSPI5_SS2	GPT_COMPARE3	PWM1_OUT	WDOG2_B	GPIO1_IO21	WDOG2_RESET_B_DEB		IN/OUT
	J21	2	132	PC21	PWM9	A0/NBS0 / PWML4	IN/OUT	E19	SD1_DAT2	SD1_DATA2	ECSPI5_SS1	GPT_COMPARE2	PWM2_OUT	WDOG1_B	GPIO1_IO19	WDOG1_RESET_B_DEB		IN/OUT
	J21	3	102	PB7	PWM10	ERXER	IN/OUT	T4	GPIO_1	ESAI_RX_CLK	WDOG2_B	KEY_ROW5	USB_OTG_ID	PWM2_OUT	GPIO1_IO01		SD1_CD_B	IN/OUT
	J21	4	20	PD7	PWM11	A17/BA1 / TIOA8	IN/OUT	T2	GPIO_9	ESAI_RX_FS	WDOG1_B	KEY_COL6	CCM_REF_EN_B	PWM1_OUT	GPIO1_IO09		SD1_WP	IN/OUT
	J21	5	21	PD8	PWM12	A21/NANDALE / TIOB8	IN/OUT	R7	GPIO_3	ESAI_RX_HF_CLK		I2C3_SCL	XTALOSC_REF_CLK_24M	CCM_CLKO2	GPIO1_IO03	USB_H1_OC	MLB_CLK	IN/OUT
	J21	6	68	PB27	PWM13	NCS3 / TIOB0	IN/OUT	D18	SD4_DAT0		SD4_DATA0	NAND_DQS			GPIO2_IO08			IN/OUT
	J21	7	GND		GND													IN/OUT
	J21	8	VDD_SNVS		AREF													IN/OUT
	J21	9	9	PA17	SDA1	TWD0 / SPCK0	IN/OUT											IN/OUT
	J21	10	70	PA18	SCL1	TWCK0 / A20 / WKUP9	IN/OUT											IN/OUT

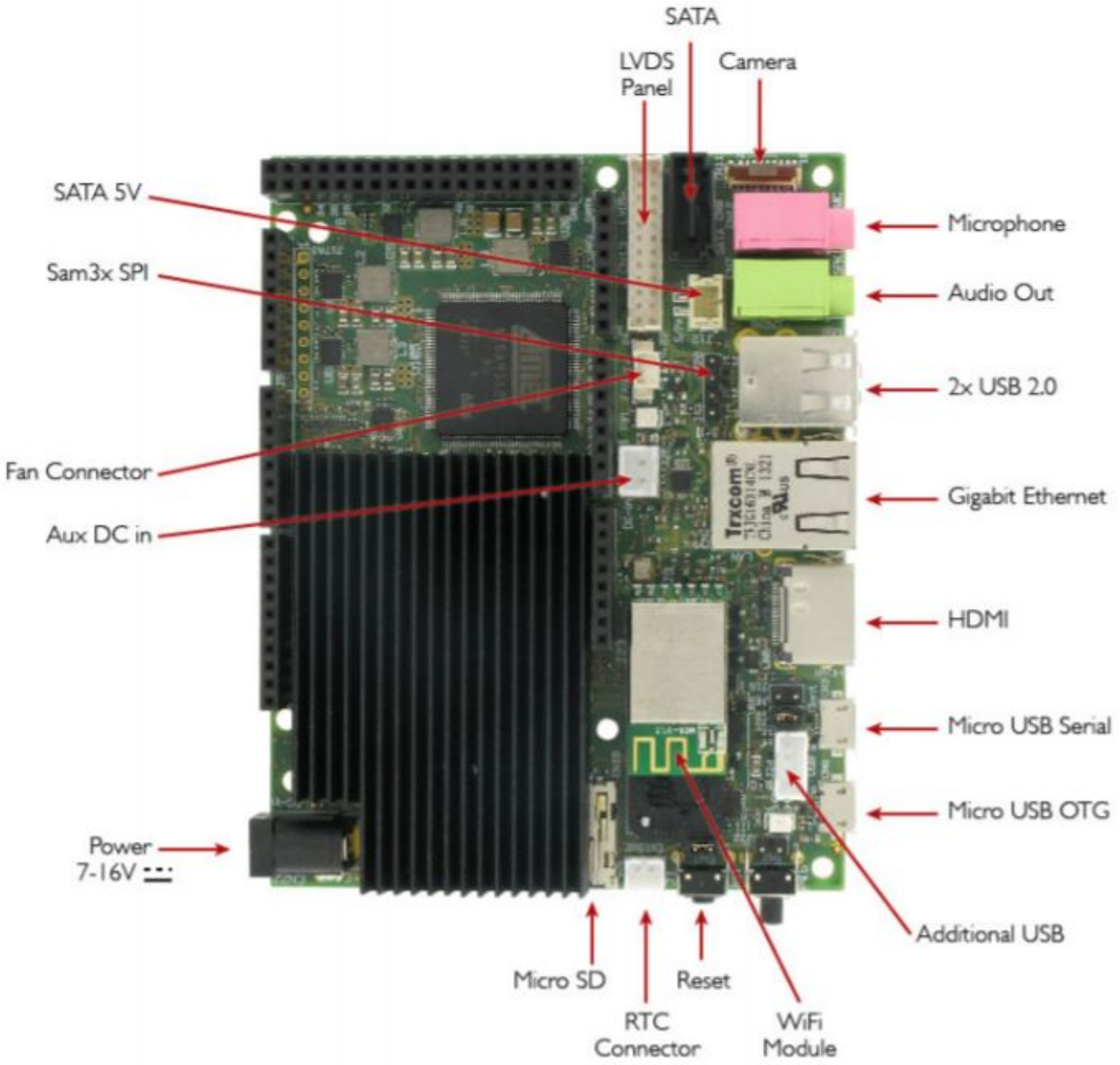
	UDOO Pin Connector		SAM3X						i.MX6										
	Bank	Physical pin	Physical pin	Pin Name	Arduino function	Possible functions	GPIO function	Physical pin	Pin Name	Pad Mode ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	GPIO function
COMUNICATION	J5	1	87	URXD/PA8	SCL	TWCK1 / PWMH1 / AD9	IN/OUT	H20	EIM_D21	EIM_DATA21	ECSPI4_SCLK	IPU1_DI0_PIN17	IPU2_CSI1_DATA11	USB_OTG_OC	GPIO3_IO21	I2C1_SCL	SPDIF_IN	IN/OUT
	J5	2	86	UTXD/PA9	SDA	TWD1 / PWMH0 / AD8	IN/OUT	G23	EIM_D28	EIM_DATA28	I2C1_SDA	ECSPI4_MOSI	IPU2_CSI1_DATA12	UART2_CTS_B	GPIO3_IO28	IPU1_EXT_TRIG	IPU1_DI0_PIN13	IN/OUT
	J5	3	3	PB25	RX1	RXD0 / DATRG / WKUP5	IN/OUT	L1	CSI0_DAT13	IPU1_CSI0_DATA13	EIM_DATA09		UART4_RX_DATA		GPIO5_IO31		ARM_TRACE10	IN/OUT
	J5	4	4	PC28	TX1	TXD0 A0TRG WKUP6	IN/OUT	M5	CSI0_DAT12	IPU1_CSI0_DATA12	EIM_DATA08		UART4_TX_DATA		GPIO5_IO30		ARM_TRACE09	IN/OUT
	J5	5	5	PA29-PC26	RX2	RXD1 / PWML1 / WKUP7	IN/OUT	M6	CSI0_DAT15	IPU1_CSI0_DATA15	EIM_DATA11		UART5_RX_DATA		GPIO6_IO01		ARM_TRACE12	IN/OUT
	J5	6	6	PC25	TX2	TXD1 / PWMH2	IN/OUT	M4	CSI0_DAT14	IPU1_CSI0_DATA14	EIM_DATA10		UART5_TX_DATA		GPIO6_IO00		ARM_TRACE11	IN/OUT
	J5	7	18	PC24	RX3	A15 / RXD3	IN/OUT	L4	CSI0_DAT16	IPU1_CSI0_DATA16	EIM_DATA12		UART4_RTS_B		GPIO6_IO02		ARM_TRACE13	IN/OUT
	J5	8	17	PC23	TX3	A14 / TXD3	IN/OUT	N1	CSI0_DAT4	IPU1_CSI0_DATA04	EIM_DATA02	ECSPI1_SCLK	KEY_COL5	AUD3_TXC	GPIO5_IO22		ARM_TRACE01	IN/OUT

	UDOO Pin Connector		SAM3X						i.MX6										
	Bank	Physical pin	Physical pin	Pin Name	Arduino function	Possible functions	GPIO function	Physical pin	Pin Name	Pad Mode ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	GPIO function
IOXL - IOXH	J8	1	VCC_SW	PB26	5V													I/O
	J8	2	VCC_SW	PA14	5V													I/O
	J8	3	1	PA15	PIN22	CTS0 / TCLK0 / WKUP15	IN/OUT	R23	DISP0_DAT6	IPU1_DISP0_DATA06	IPU2_DISP0_DATA06	ECSPI3_SS3	AUD6_RXC			GPIO4_IO27		I/O
	J8	4	7	PD0	PIN23	RTS1 / TK	IN/OUT	R24	DISP0_DAT7	IPU1_DISP0_DATA07	IPU2_DISP0_DATA07	ECSPI3_RDY				GPIO4_IO28		I/O
	J8	5	8	PD1	PIN24	CTS1 / TF / WKUP8	IN/OUT	R22	DISP0_DAT8	IPU1_DISP0_DATA08	IPU2_DISP0_DATA08	PWM1_OUT	WDOG1_B			GPIO4_IO29		I/O
	J8	6	13	PD2	PIN25	A10 / RXD4	IN/OUT	T25	DISP0_DAT9	IPU1_DISP0_DATA09	IPU2_DISP0_DATA09	PWM2_OUT	WDOG2_B			GPIO4_IO30		I/O
	J8	7	14	PD3	PIN26	A11 / MCDA5	IN/OUT	R21	DISP0_DAT10	IPU1_DISP0_DATA10	IPU2_DISP0_DATA10					GPIO4_IO31		I/O
	J8	8	15	PD6	PIN27	A12/ MCDA6	IN/OUT	T23	DISP0_DAT11	IPU1_DISP0_DATA11	IPU2_DISP0_DATA11					GPIO5_IO05		I/O
	J8	9	16	PD9	PIN28	A13 / MCDA7	IN/OUT	T24	DISP0_DAT12	IPU1_DISP0_DATA12	IPU2_DISP0_DATA12					GPIO5_IO06		I/O
	J8	10	19	PA7	PIN29	A16/BA0 / PWMFI2	IN/OUT	R20	DISP0_DAT13	IPU1_DISP0_DATA13	IPU2_DISP0_DATA13		AUD5_RXFS			GPIO5_IO07		I/O
	J8	11	22	PD10	PIN30	A22/NANDCLE / TCLK8	IN/OUT	U25	DISP0_DAT14	IPU1_DISP0_DATA14	IPU2_DISP0_DATA14		AUD5_RXC			GPIO5_IO08		I/O
	J8	12	26	PC1	PIN31	TCLK2 / NCS1 / WKUP3	IN/OUT	T22	DISP0_DAT15	IPU1_DISP0_DATA15	IPU2_DISP0_DATA15	ECSPI1_SS1	ECSPI2_SS1			GPIO5_IO09		I/O
	J8	13	32	PC2	PIN32	NWR1/NBS1	IN/OUT	T21	DISP0_DAT16	IPU1_DISP0_DATA16	IPU2_DISP0_DATA16	ECSPI2_MOSI	AUD5_TXC	SDMA_EXT_EVENT0		GPIO5_IO10		I/O
	J8	14	55	PC3	PIN33		IN/OUT	U24	DISP0_DAT17	IPU1_DISP0_DATA17	IPU2_DISP0_DATA17	ECSPI2_MISO	AUD5_TXD	SDMA_EXT_EVENT1		GPIO5_IO11		I/O
	J8	15	59	PC4	PIN34	D0 / PWML0	IN/OUT	V25	DISP0_DAT18	IPU1_DISP0_DATA18	IPU2_DISP0_DATA18	ECSPI2_SS0	AUD5_TXFS	AUD4_RXFS		GPIO5_IO12	EIM_CS2	I/O
	J8	16	60	PC5	PIN35	D1 / PWMH0	IN/OUT	U23	DISP0_DAT19	IPU1_DISP0_DATA19	IPU2_DISP0_DATA19	ECSPI2_SCLK	AUD5_RXD	AUD4_RXC		GPIO5_IO13	EIM_CS3	I/O
	J8	17	116	PC6	PIN36	D2 / PWML1	IN/OUT	U22	DISP0_DAT20	IPU1_DISP0_DATA20	IPU2_DISP0_DATA20	ECSPI1_SCLK	AUD4_TXC			GPIO5_IO14		I/O
	J8	18	63	PC7	PIN37	D3 / PWMH1	IN/OUT	T20	DISP0_DAT21	IPU1_DISP0_DATA21	IPU2_DISP0_DATA21	ECSPI1_MOSI	AUD4_TXD			GPIO5_IO15		I/O
	J8	19	64	PC8	PIN38	D4 / PWML2	IN/OUT	H25	EIM_A16	EIM_ADDR16	IPU1_DI1_DISP_CLK	IPU2_CSI1_PIXCLK				GPIO2_IO22		I/O
	J8	20	65	PC9	PIN39	D5 / PWMH2	IN/OUT	P6	GPIO_18	ESAI_TX1	ENET_RX_CLK	SD3_VSELECT	SDMA_EXT_EVENT1			GPIO7_IO13	SNVS_VIO_5_CTL SRC_BOOT_CFG14	I/O
	J8	21	66	PA19	PIN40	D6 / PWML3	IN/OUT	A18	NANDF_D0	NAND_DATA04	SD1_DATA4					GPIO2_IO00		I/O
	J8	22	67	PA20	PIN41	D7 / PWMH3	IN/OUT	D17	NANDF_D3	NAND_DATA03	SD1_DATA7					GPIO2_IO03		I/O
	J8	23	71	PC19	PIN42	MCCK / PWMH1	IN/OUT	F16	NANDF_D2	NAND_DATA02	SD1_DATA6					GPIO2_IO02		I/O
	J8	24	72	PC18	PIN43	MCCDA / PWML2	IN/OUT	C17	NANDF_D1	NAND_DATA01	SD1_DATA5					GPIO2_IO01		I/O
	J8	25	101	PC17	PIN44	NANDOE / PWMH5	IN/OUT	P5	GPIO_19	KEY_COL5	ENET_1588_EVENT0_OUT	SPDIF_OUT	CCM_CLKO1	ECSPI1_RDY		GPIO4_IO05	ENET_TX_ER	I/O
	J8	26	100	PC16	PIN45	NWR0/NWE / PWMH6	IN/OUT	V24	DISP0_DAT22	IPU1_DISP0_DATA22	IPU2_DISP0_DATA22	ECSPI1_MISO	AUD4_TXFS			GPIO5_IO16		I/O
	J8	27	99	PC15	PIN46	D15 / ETXER	IN/OUT	W24	DISP0_DAT23	IPU1_DISP0_DATA23	IPU2_DISP0_DATA23	ECSPI1_SS0	AUD4_RXD			GPIO5_IO17		I/O
	J8	28	98	PC14	PIN47	D14 / ETX3	IN/OUT	G22	EIM_D25	EIM_DATA25	ECSPI4_SS3	UART3_RX_DATA	4ECSPI1_SS3	ECSPI2_SS3		GPIO3_IO25	AUD5_RXC	I/O
	J8	29	97	PC13	PIN48	D13 / ETX2	IN/OUT	U6	KEY_ROW1	ECSPI1_SS0	ENET_COL	AUD5_RXD	KEY_ROW1	UART5_RX_DATA		GPIO4_IO09	SI2_VSELECT	UART1_DSR_B I/O
	J8	30	96	PC12	PIN49	D12 / ERXCK	IN/OUT	U7	KEY_COL1	ECSPI1_MISO	ENET_MDIO	AUD5_TXFS	KEY_COL1	UART5_TX_DATA		GPIO4_IO08	SD1_VSELECT	I/O
	J8	31	95	PB21	PIN50	D11 / ECOL	IN/OUT	J24	EIM_OE	EIM_OE	IPU1_DI1_PIN07	ECSPI2_MISO				GPIO2_IO25		I/O
	J8	32	94	PB14	PIN51	D10 / ERX3	IN/OUT	J23	EIM_CS1	EIM_CS1	IPU1_DI1_PIN06	ECSPI2_MOSI				GPIO2_IO24		I/O
	J8	33	92		PIN52	RXD2 / SPI0_NPCS2 / AD14/WKUP13	IN/OUT	H24	EIM_CS0	EIM_CS0	IPU1_DI1_PIN05	ECSPI2_SCLK				GPIO2_IO23		
	J8	34	140		PIN53	CANTX1 / PWMH2	IN/OUT	F22	EIM_D24	EIM_DATA24	ECSPI4_SS2	UART3_TX_DATA	ECSPI1_SS2	ECSPI2_SS2		GPIO3_IO24	AUD5_RXFS	UART1_DTR_B
	J8	35	GND		GND													
	J8	36	GND		GND													

	UDOO Pin Connector		SAM3X						i.MX6										
	Bank	Physical pin	Physical pin	Pin Name	Arduino function	Possible functions	GPIO function	Physical pin	Pin Name	Pad Mode ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	GPIO function
POWER	J23	1			NOT USED													
	J23	2	VDD_SNVS		IOREF													
	J23	3	47		NRSTB	RESET												
	J23	4	VDD_SNVS		3.3V													
	J23	5	VCC_SW		5V													
	J23	6	GND		GND													
	J23	7	GND		GND													
	J23	8	VIN		VIN													
ADCL	J11	1	78	PA16	A0	SPCK1 / TD / AD7	IN/OUT											
	J11	2	79	PA24	A1	MCDA3 / PCK1 / AD6	IN/OUT											
	J11	3	80	PA23	A2	MCDA2 / TCLK4 / AD5	IN/OUT											
	J11	4	81	PA22	A3	MCDA1 / TCLK3 / AD4	IN/OUT											
	J11	5	82	PA6	A4	TIOB2 / NCS0 / AD3	IN/OUT											
	J11	6	83	PA4	A5	TCLK1 / NWAIT / AD2	IN/OUT											
	J11	7	84	PA3	A6	TIOB1 / PWMFI1 / AD1/WKUP1	IN/OUT											
	J11	8	85	PA2	A7	TIOA1 / NANDRDY / AD0	IN/OUT											
ADCH	J15	1	88	PB17	A8	RF / PWML1 / AD10	IN/OUT											
	J15	2	89	PB18	A9	RD / PWML2 / AD11	IN/OUT											
	J15	3	90	PB19	A10	RK / PWML3 / AD12	IN/OUT											
	J15	4	91	PB20	A11	TXD2 / SPI0_NPCS1 / AD13	IN/OUT											
	J15	5	76	PB15	DAC0	CANRX1 / PWMH3 /DAC0/WKUP12	IN/OUT											
	J15	6	77	PB16	DAC1	TCLK5 / PWML0 / DAC1	IN/OUT											
	J15	7	23	PA0	CANRX	CANTX0 / PWML3	IN/OUT	R3	GPIO_7	ESAI_TX4_RX1	ECSPI5_RDY	EPIT1_OUT	FLEXCAN1_TX	UART2_TX_DATA	GPIO1_IO07	SPDIF_LOCK	USB_OTG_HOST_MODE	IN/OUT
	J15	8	24	PA1	CANTX	CANRX0 / PCK0 / WKUP0	IN/OUT	R5	GPIO_8	ESAI_TX5_RX0	XTALOSC_REF_CLK_32K	EPIT2_OUT	FLEXCAN1_RX	UART2_RX_DATA	GPIO1_IO08	SPDIF_SR_CLK	USB_OTG_PWR_CTL_WAKE	IN/OUT

SECO USA Inc | 185 Devonshire St, 02110 Boston, MA
SECO Srl | Via Calamandrei 91, Arezzo, Italy P.IVA 00325250512
email:info@udoo.org

Figure C.3: UDOO pinout - detailed version


SATA
LVDS Panel
Camera
SATA 5V
Sam3x SPI
Fan Connector
Aux DC in
Microphone
Audio Out
2x USB 2.0
Gigabit Ethernet
HDMI
Micro USB Serial
Micro USB OTG
Power 7-16V
Additional USB
Micro SD
RTC Connector
Reset
WiFi Module