

TOUCHDESIGNER
INTERACTIVITY COURSE
CLASS I –
ARDUINO DISTANCE SENSOR

GOAL OF THE COURSE

During these classes we'll focus on interactivity within Touchdesigner. We'll try to get acquainted with different sensors and how they work.

We'll combine generative sound with TouchDesigner, where parameters will control certain filters or sounds

Arduino will make it's introduction into these classes where a sensor can control a visual or Touchdesigner can control a motor through Arduino.

THE DIFFERENT KIND OF SENSORS

Leap Motion – available for mac and windows, a hand gesture sensor


Xbox Kinect – available only for windows, a full body skeleton tracking sensor

Webcam – integrated into your own device we'll utilize it to do blobtracking and movement detection

Entec – not a real sensor, DMX to usb controller to relay DMX messages to peripherals like lighting

GENERATIVE SOUND

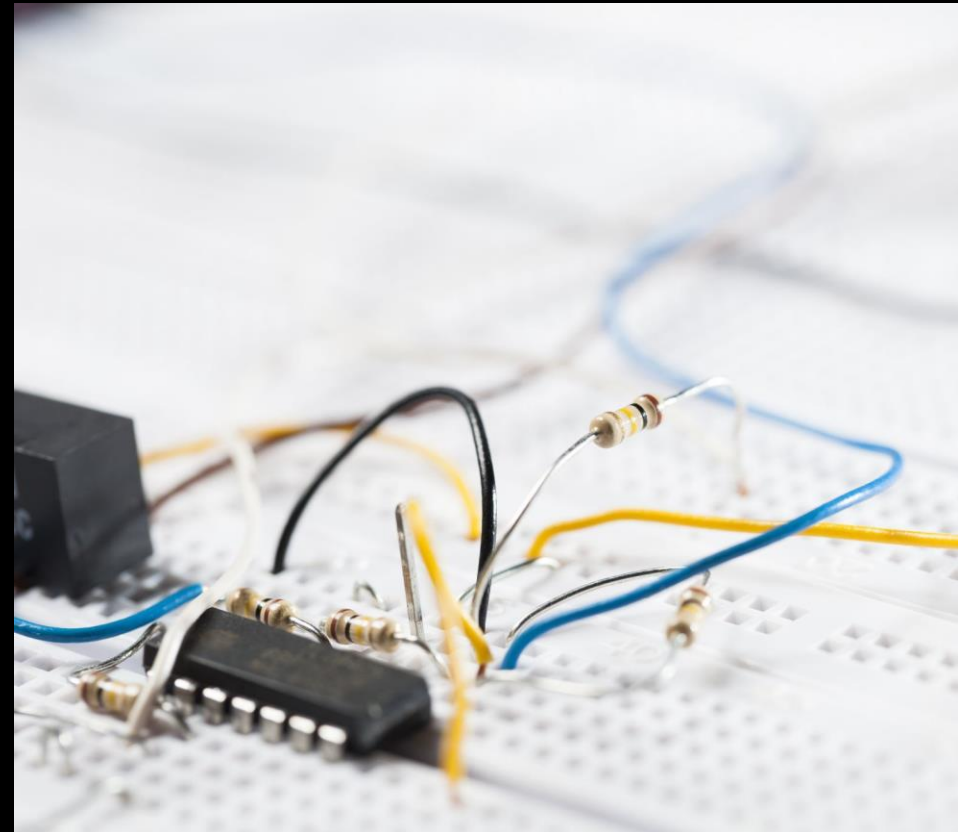
We're going to take a look at PureData, an open source graphical programming environment like Touchdesigner specialized towards sound. It's more expensive but precisely the same counterpart would be MAX/MSP. An environment that allows you to build synthesizers and effects through connecting nodes. We will mostly work with premade patches to streamline workflow.



When in demand we can also look at Ableton's integration into Touchdesigner to use parametrization as MIDI or effect control.

ARDUINO

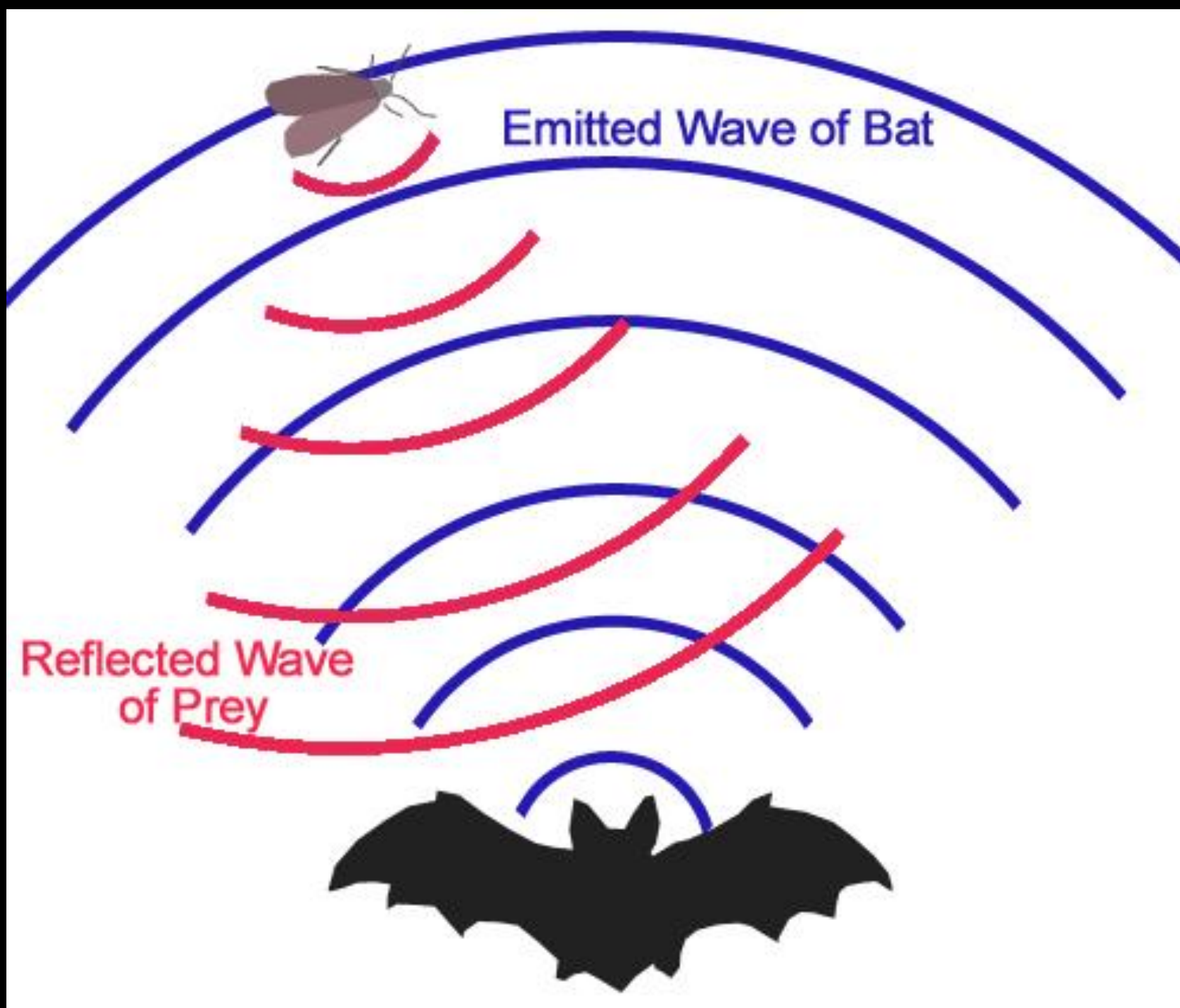
- We'll use preprogrammed arduino's with several different sensors and import the raw sensor data into Touchdesigner to process it further. Once the workflow is known it is basically applicable to all Arduino projects that work on a similar level.
- We'll also use Arduino as an importer/translator that will receive data from Touchdesigner so we can use parametrization as control for servo's, led's or other peripherals.

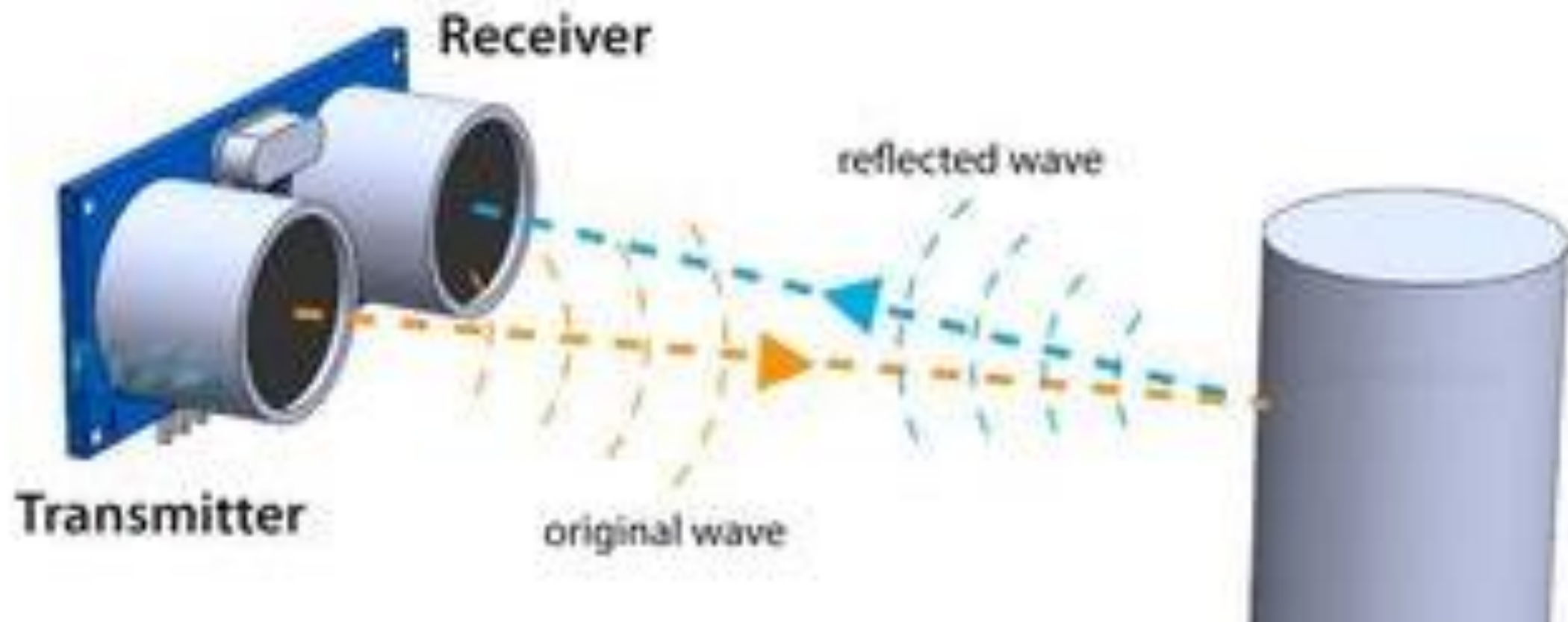


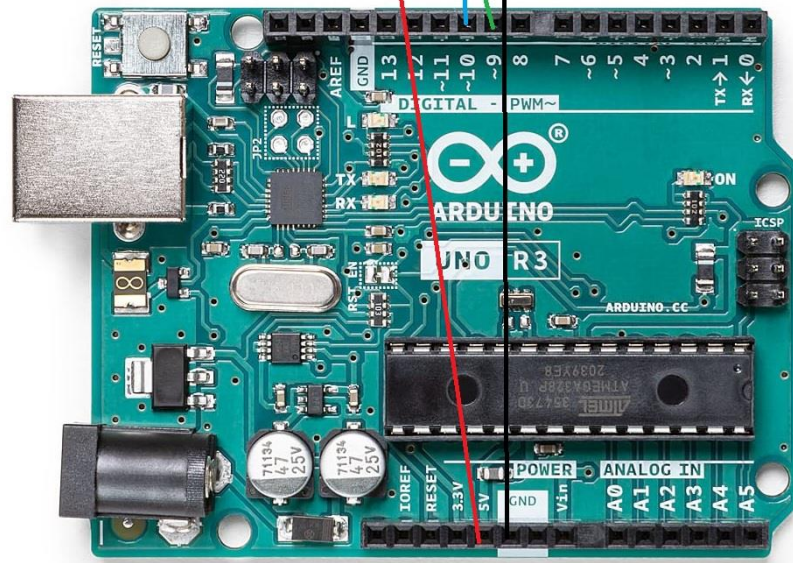
ULTRASONIC DISTANCE SENSOR

This week we'll be working with the ultrasonic distance sensor connected to Arduino. The sensor can register the distance in relation to itself and other obstacles through the time of flight principle. Pretty much the same as the hearing of a bat works.



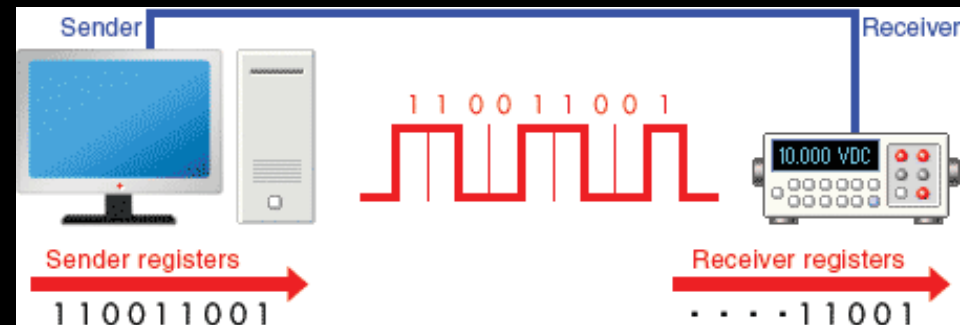






SERIAL COMMUNICATION

- The arduino communicates to the computer, or can communicate with the computer, through a protocol which is called *Serial Communication*. I will not go into detail too much but there are some things you need to know.
- Serial communication sends information over a dataline one bit at the time. Therefore it is fast and reliable on a short distance but it can also only be communicating with *one device per time*.



ASCII TO BINARY AND BACK AGAIN

- From Arduino we are sending string-values, meaning that we send text like the following: "Distance: 120 cm". In computer speak any kind of text is called a string and a string usually is formatted in ASCII, American Standard Code for Information Interchange, meaning that every symbol in the alphabet, which is human readable, is also represented by a binary counterpart which is computer readable again.
- Since Serial sends one bit per time all the text that we will send has to be converted from ASCII back into binary. In a lot of other programs receiving serial data you have to do this conversion 'manually', lucky enough Touchdesigner handles this for us we do not need to worry about translating binary back into ASCII.

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

TouchDesigner 2023.11880: C:/Users/Admin/Desktop/NewProject.1.toe*

File Edit Dialogs Help

WIKI FORUM TUTORIALS OI 60 FPS: 60 Realtime 13:40:59 Operator serial1 deleted. UPDATE

Pane Layout

New Layout +

/ project1

0

0 message

1 Distance: 61.28

2 Distance: 61.38

3 Distance: 59.96

4 Distance: 59.85

5 Distance: 59.54

serial1

0

0 message

1 Distance: 61.28

2 Distance: 61.38

3 Distance: 59.96

4 Distance: 59.85

5 Distance: 59.54

convert1

0

0 61.28

select1

61.28 chan1

datto1

1 # me - this DAT

2 #

3 # dat - the DAT tha

4 # rowIndex - the row

5 # message - an asci

6 # Unprint

7 # not be

8 # the raw

serial1_callbacks

Start: 1 End: 600

RStart: 1 REnd: 600

FPS: 60.0 Tempo: 120.0

ResetF: 1 T Sig: 4 4

TimeCode

00:00:01:54

115

Range Limit

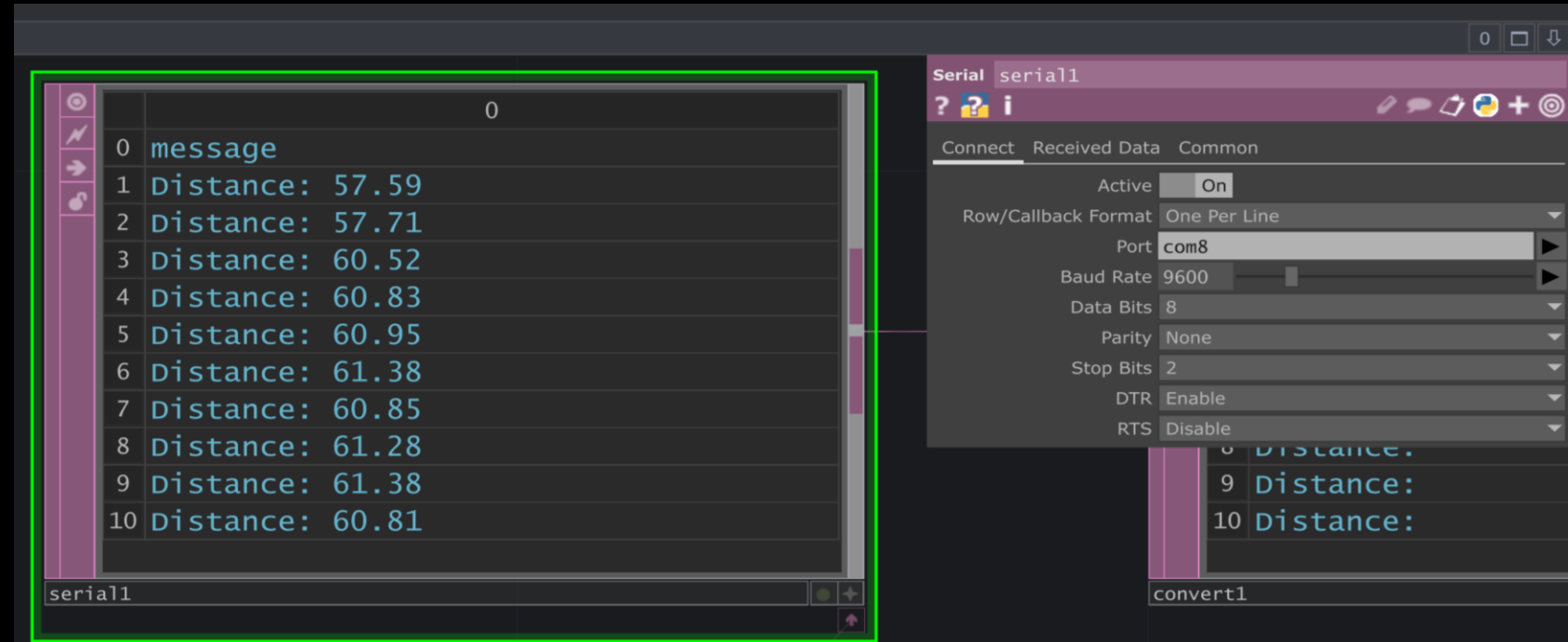
Loop Once

SERIAL DAT

First we'll place a Serial DAT to import the serial data. The reason why we use a DAT in this case is so we can access the string based data directly. If you would use a Serial CHOP you would have to convert decimal back into ASCII again.

Note the parameter window where it says PORT and Baud Rate. The port references the physical connection between the computer and the Arduino.

You can use the dropdown menu to select. Baud Rate refers to the speed of communication. By default this is usually 9600.



CONVERT DAT

Before we can use the numerical data coming in through the Serial DAT we need to separate the values from the word and sign "Distance: ". We can do that with a convert which can separate characters or symbols into different cell through a specific divider. "Distance:" and the values are seperated by a space so we want that to be our divider. In the parameter window input " ", so quotation marks followed by a space and then another quotation marks, at the 'Split cells at' fieldbox.

The screenshot displays a data processing workflow. On the left, a table labeled 'convert1' contains the following data:

	0	1
0	message	
1	Distance:	60.83
2	Distance:	60.83
3	Distance:	61.36
4	Distance:	61.38
5	Distance:	60.40
6	Distance:	60.83
7	Distance:	60.52
8	Distance:	60.95
9	Distance:	60.85
10	Distance:	60.85

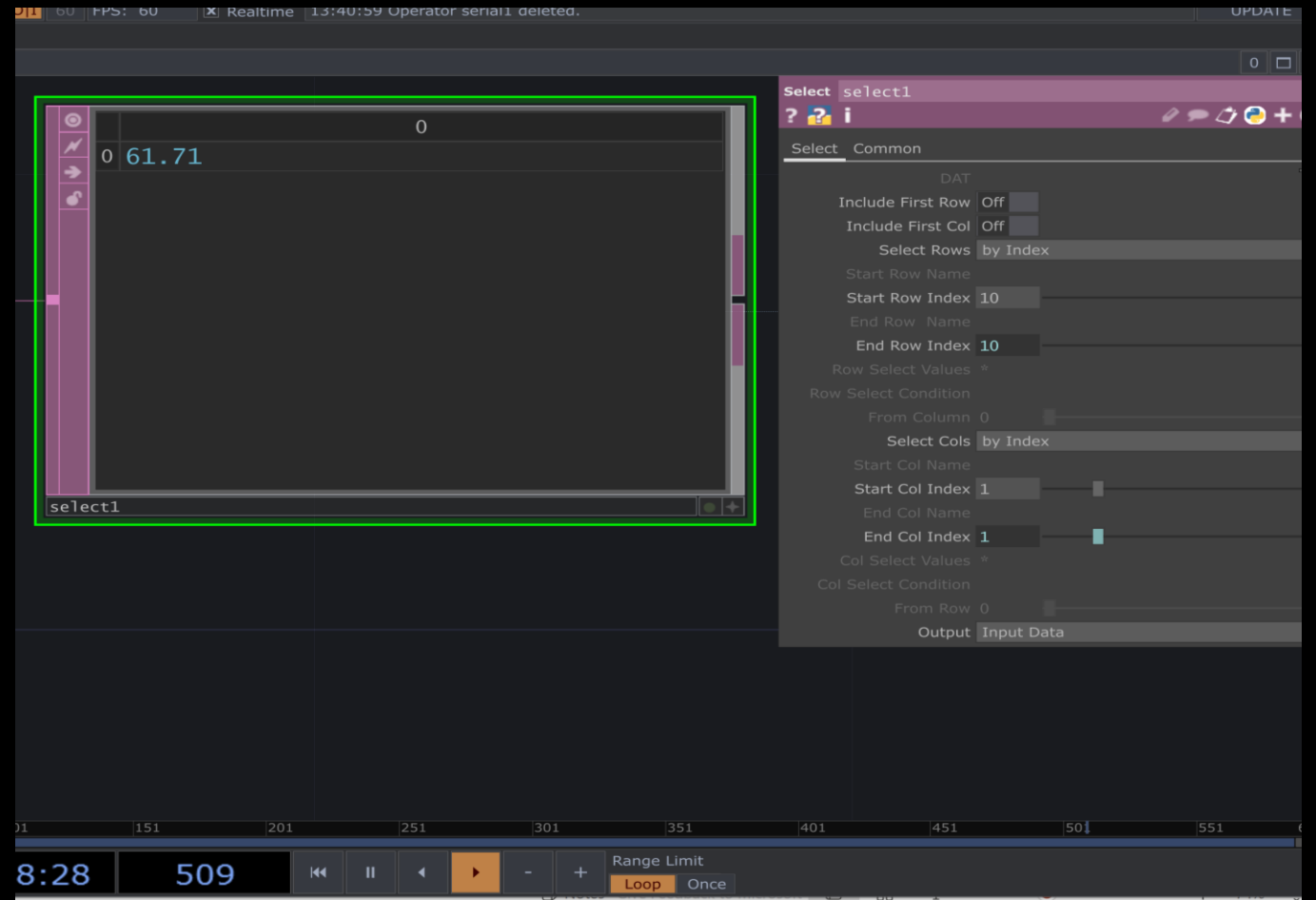
On the right, the 'Convert' block configuration window is shown. The 'Split Cells at' field is set to " ", and the 'Concatenate with' field is set to \t. The 'Remove Blank Lines' checkbox is unchecked.

SELECT DAT

When you look closely at the values coming in at Serial DAT you can see that line 10 is the newest value and then travels up. For the sake of latency reduction we want to access the most updated value first, which would be row 10.

Also, because we separated the 'Distance: ' from the values we want to only access column 1 and omit column 0 because it holds no relevant information for us.

We can do this easily with a select DAT. We can choose to select rows and columns by index. For the index of rows we want to start and end at 10, the most updated value. And for the columns we want to start and end at 1, containing the values.



DAT TO CHOP

Lastly what we want to do is to connect the Select DAT to a DATtoCHOP. We want to do this so the values that live inside the DAT operators, which are string, or text, based are converted back into numbers again.

If you would try to reference the numbers inside the DAT operators, which are basically text, onto a number box parameter throughout your network you will get errors complaining how text is being referenced instead of values.

On the parameter window make sure that you select that first row and first columns are values.

The screenshot displays a software interface with a central workspace and a right-hand parameter window. The workspace contains a DAT operator labeled 'datto1' at the bottom left, which is highlighted with a green border. A vertical blue bar is positioned in the center of the workspace, with the text '59.12 chan1' next to it. The parameter window on the right is titled 'DAT to datto1' and contains a 'Common' tab. Under the 'DAT' section, 'select1' is selected. The 'Select Rows' section is set to 'All', and the 'Select Cols' section is also set to 'All'. The 'Output' is set to 'Channel per Row', 'First Row is' is 'Values', 'First Column is' is 'Values', 'Default Value' is '0', and 'Duplicate Names' is 'Make Unique'. At the bottom of the interface, there is a timeline with markers at 101, 151, 201, 251, 301, 351, 401, 451, 501, and 551. A blue bar at the bottom shows a time of 01:05, a value of 66, and playback controls including a range limit and a loop button.

EASY PEASY LEMON BREEZY

- After this relative simple network chain you can access your sensorical values being sent from Arduino to Touchdesigner. You can process the values further through other CHOP's or use it directly to influence other parameters or values throughout your network.
- You could use a LOGIC CHOP to detect wether or not you are within a certain range and have it activate things. You could use it in conjunction with MATH CHOP to rescale the values and have the sensor control parts of a video. Or you could use a COUNT CHOP in conjunction with a LOGIC CHOP to count the amount of people or obstacles passing the sensor changing either video, light or sound within your animation. Or perhaps trigger different scenes.
- With this very basic setup you can already do a lot of different interactive things.

BUILD YOUR OWN AND PRESENT THE NETWORK

- We are going to take X-amount of time to build a tabletop installation in class that uses the sensor in 3 different ways simultaneously.
- We'll start at 11:15 to look at eachothers work and how everyone used their sensor or handled the data.
- You may work alone or in small groups of 3.