TUTORIAL 3 – ADVANCED FLOW CONTROL

EQUIPMENT NEEDED

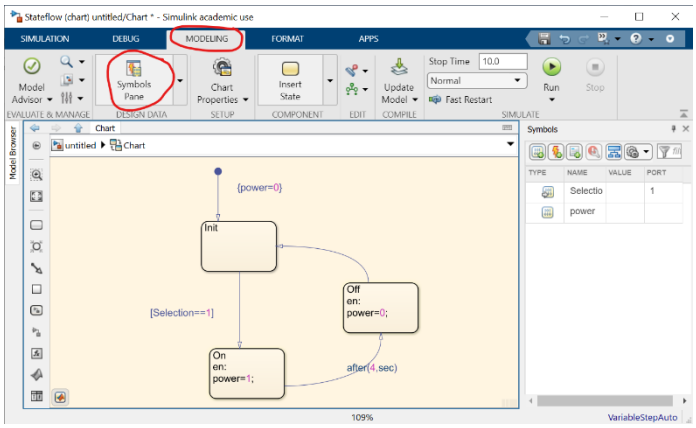
Your Laptop
Arduino Nano 33 IoT
PC-based Oscilloscope + Probes
(Optional) Breadboard
(Optional) LEDs, Switches, etc

FINITE STATE MACHINES

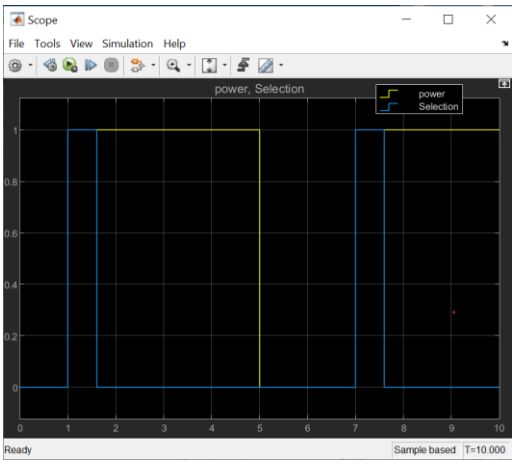
SIMPLE STATE MACHINE

1. Using the block ***Chart*** (Stateflow) create the state flow diagram showed below.

> Note: In order to see all symbols in the chart (i.e. *power, Selection*), as well as messages and events later, and change their behaviour (input, output, local) you will have to open the symbols sidebar via the tab *Modelling* and selecting the *Symbols Pane*



2. Connect a *Source* (you can decide which one) to the input port of the chart. It needs to send one or more 1s to start the state change.
3. Connect a ***Scope*** to the output as well as the *Source* to monitor the power and Selection.

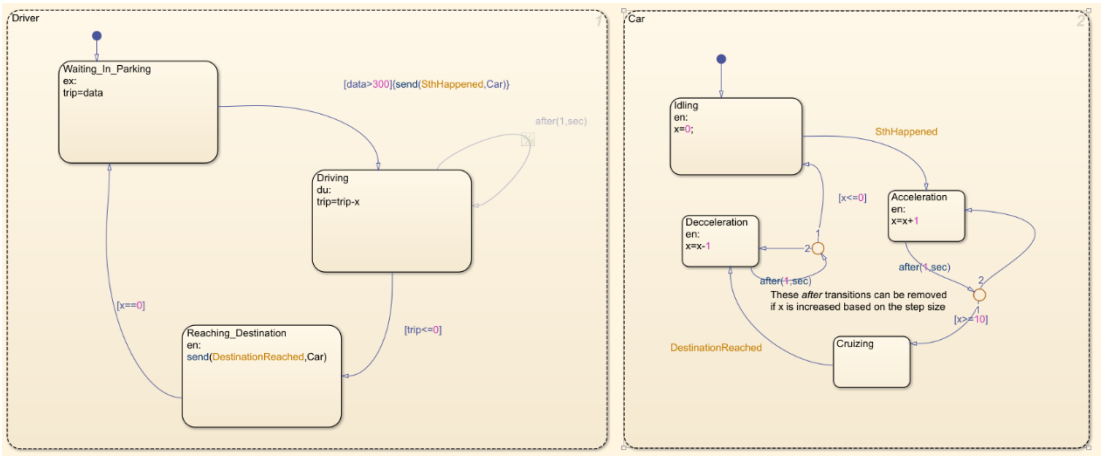
> The output you are looking for can be seen below. I used a ***Pulse Generator*** (Period:6, Pulse Width:10%, Phase delay:1. *Stop time* is 10.
>
> 
>

4. Try to replicate the same chart with less states. (Hint: Use transition actions)
5. Experiment with more states in the chart, more outputs, more inputs.

6. Create a state machine that combines ***WiFi UDP (or TCP/IP) Receive*** blocks with ***Digital Output*** blocks. For example, based on the state of the machine you can alter the intensity of an LED.

STATE MACHINE EVENTS AND MULTIPLE SUBSTATES

As a second task you want to recreate the Driver-Car state machine from the lecture. An image of it below:



7. Start by creating the two main ***States*** and then put the smaller ones inside.
8. To enable the dashed border for the *Driver* and *Car* states you need to make them to operate in *Parallel*. To do so on the empty canvas *<right click>->Decomposition->Parallel (AND)*.
9. From the *Modeling* tab, in *Symbols Pane* ensure that the symbols are defined as:

Type	Name
Input Data	data
Local Data	x
Output Data	trip
Internal Event	SthHappened
Internal Event	DestinationReached
State	Car

10. Connect to the input port (*data*) a ***Random Number*** block that produces values about 300 occasionally (e.g. *Mean: 280, Variance: 150, Seed:0*)
11. Connect the output port (*trip*) to a ***Scope***.
12. Add a ***Sequence Viewer*** block to the model. After you run the model you can see how the states and the events moved between the *Driver* and the *Car*. (This can be used with multiple State machines on the same Model too).
13. (optional) You can use a ***Relational Operator*** to monitor when the value goes above 300 externally to the State Machine.

FUNCTIONS AND EVENTS

For some of the actions below when you try to combine different subsystems you will need to use the ***Merge*** block for the outputs to work properly.

FUNCTION-CALL SUBSYSTEMS

1. Create two ***Function-call subsystem*** that are altering a pin or are sending data back via a ***WiFi UDP Send*** block.
2. Pay attention on the *Block Parameters* of the ***Outport*** of the subsystems. Ensure that the property *Output when disabled* is set to *reset* and that the *Initial output* is 0
3. Try different ways to call these subsystems. Try a ***Function-call generator***, try an *Event* from a Stateflow chart.
4. Experiment with the ***Function-call split*** block.

> Note: This block is most relevant when you need to ensure the sequence of your functions. Also, this block functions closely with the **Function-call feedback Latch**. To understand its function, type the following command in Matlab *doc 'Function-Call Feedback Latch'* and open and run the example model *openExample('simulink/FunctionCallFeedbackLatchWithSplitBlockExample')*

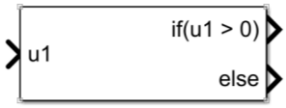
DIFFERENCE BETWEEN TRIGGERED AND ENABLED SUBSYSTEMS

1. Create an ***Enabled Subsystem*** and a ***Triggered Subsystem***.
2. Make the subsystems to either sent a value out to a ***Scope*** or do something physically with the Arduino.
3. Using the ***Signal Builder*** create a pulse that is longer than your simulation step, i.e. 2 or 3 seconds.
4. Run the model and start experimenting with the *trigger type* of the **Trigger** block in the ***Triggered Subsystem***. Notice the different behaviours.

> Note: We have seen those subsystems in the Communications tutorial where we used them to create the setup and read phases for the sensors.

(OPTIONAL) IF BLOCKS AND CASE BLOCKS

You might have used in previous tutorials the **if** block:



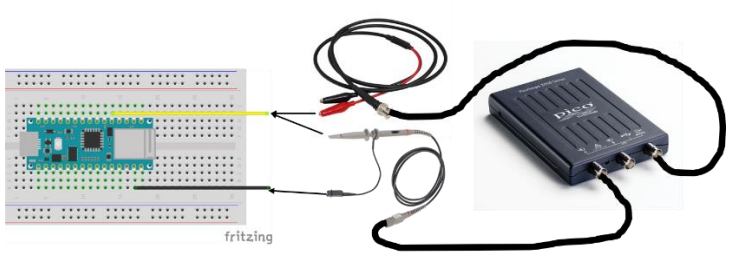
The output of suck block is in effect a *function-called* subsystem. It is a crude way of achieving some of the simplere behaviours that a State Machine chart can achieve. As a test you can do the following:

1. Connect a ***WiFi Receive*** block to an ***if*** block.
2. Create a number of ***if action subsystems*** that will turn on LEDs.
3. Create a number of ***if action subsystems*** that will change the dim (PWM) an LED.

Try the same set of actions with the ***Switch Case*** blocks.

HARDWARE INTERRUPTS

We are going to implement an *Interrupt Service Routine* for one of the *digital pins* of the Arduino. This is extremely useful for the next tutorial where we want to read the Encoder pulse train. You need to wire the PC-based oscilloscope as follows:



1. Use the ***External Interrupt*** block from the Arduino Library.
2. Set the pin to one of the interrupt enabled pins (click the *View Pinmap* button in parameters). The above connection diagram is using pin 2.
3. Leave unchecked the *Pull-Up input pin.* Leave the *Mode* set to *Rising* for now.
4. Connect the *IRQ* port to a ***Function Subsystem.***
5. In the *subsystem* use blocks to increase a value by one every time the subsystem runs (Hint: you need to use the ***Memory*** block)

Note: For any block in the subsystem you need to set their sample time to **inherit** or **-1**

6. Add a ***Rate transition*** block to the output. Be sure to untick the option *Ensure deterministic data transfer.*
7. Connect a ***Sink*** block after the transition block. A ***Display*** will work best. This will show you the number of pulses measured.
8. Via the *Solver Information* and the *Solver Settings* set the *Fixed-step size* to 0.1 (i.e. 100msec).
9. How can you calculate the frequency? Use one or more blocks to do that. You will need to take the signal from the *transition* block (Hint: it can be done with a single block. What is the frequency with respect to the pulses?). Connect the result to a ***Display***.

We are going to use the Oscilloscopes Wave Gen functionality for a more regular signal.

SETUP THE WAVE GENERATOR BEFORE YOU TURN THE SIGNAL ON

In the PicoScope software setup the generator to produce a:

Waveform: Square, Start Frequency: 20Hz, Amplitude: 1V, Offset: 1V

10. Tick the *Signal On*
11. In Simulink: Start the model via the *Hardare* tab with *Monitor & Tune*
12. When it completes compiling and downloading you should see in the pulses display the number increasing and, in the frequency display the number 20 (or very close to it)
13. In Picoscope: Increase the *Frequecny* and observe In Simulink the frequency ***Display*** following.
14. In Picoscope: Decrease the *Frequency*. At what Frequency does the ***Display*** shows occasionally 0s? Why? How can you fix it? (Hint: We made a step above that influences this)

Hint: The model in Simulink should look something like this: