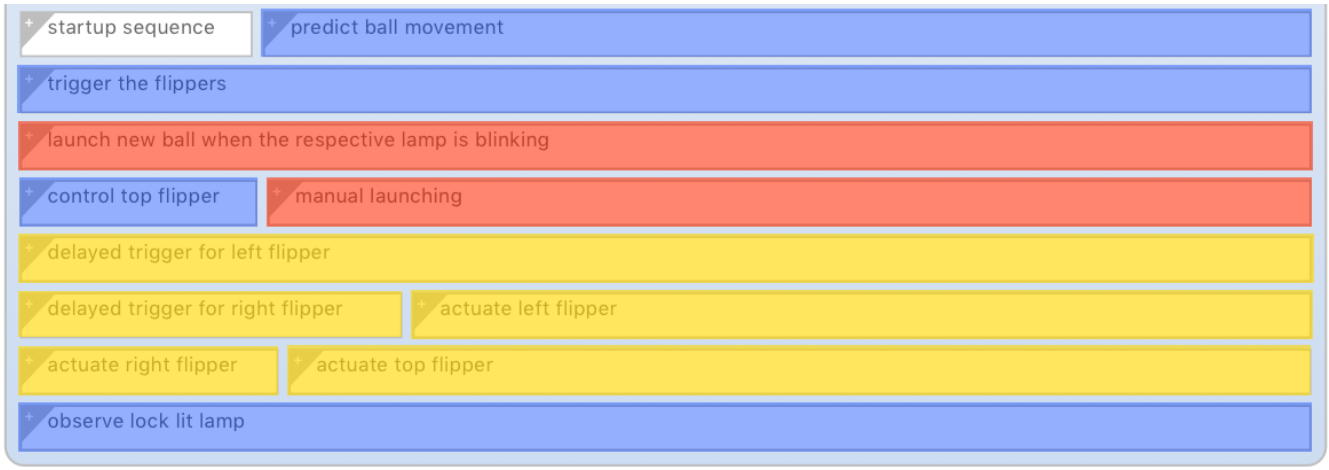
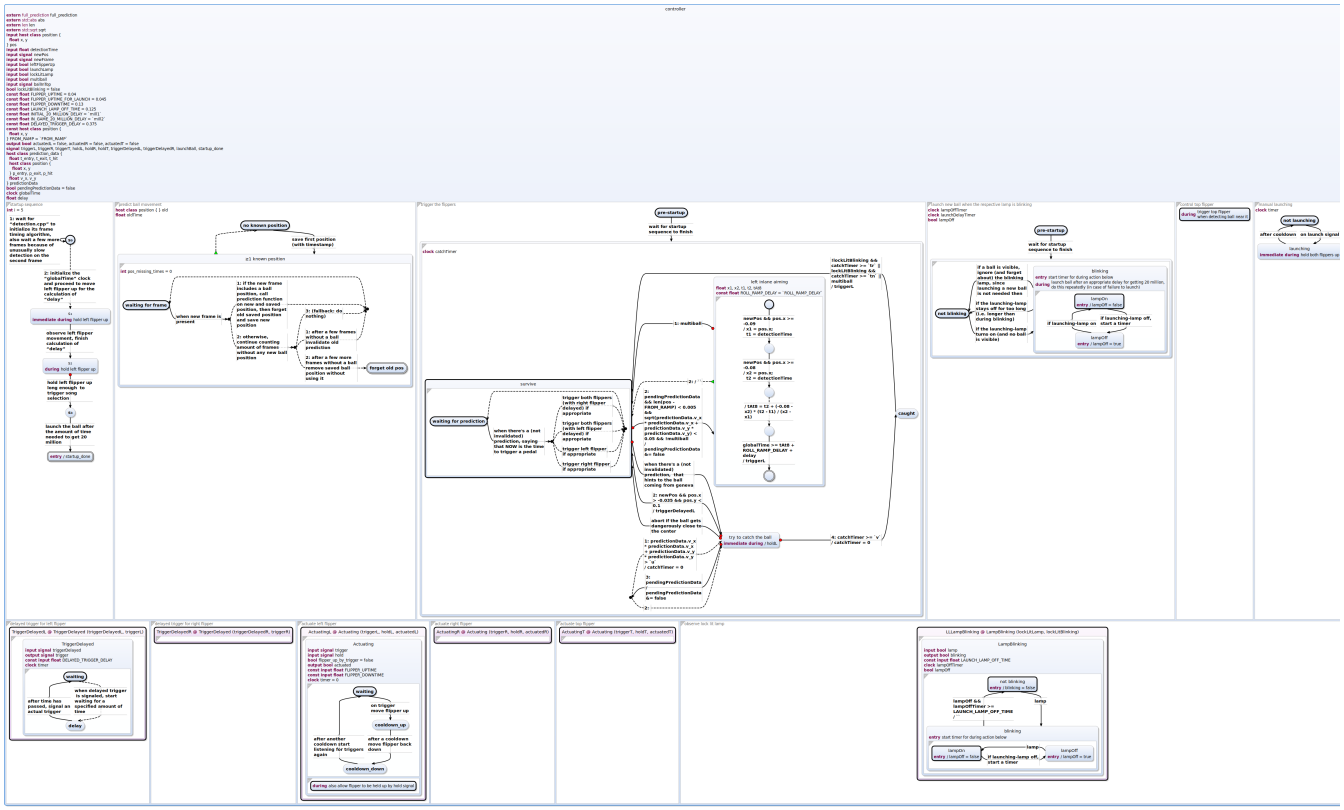
SCChart Controller

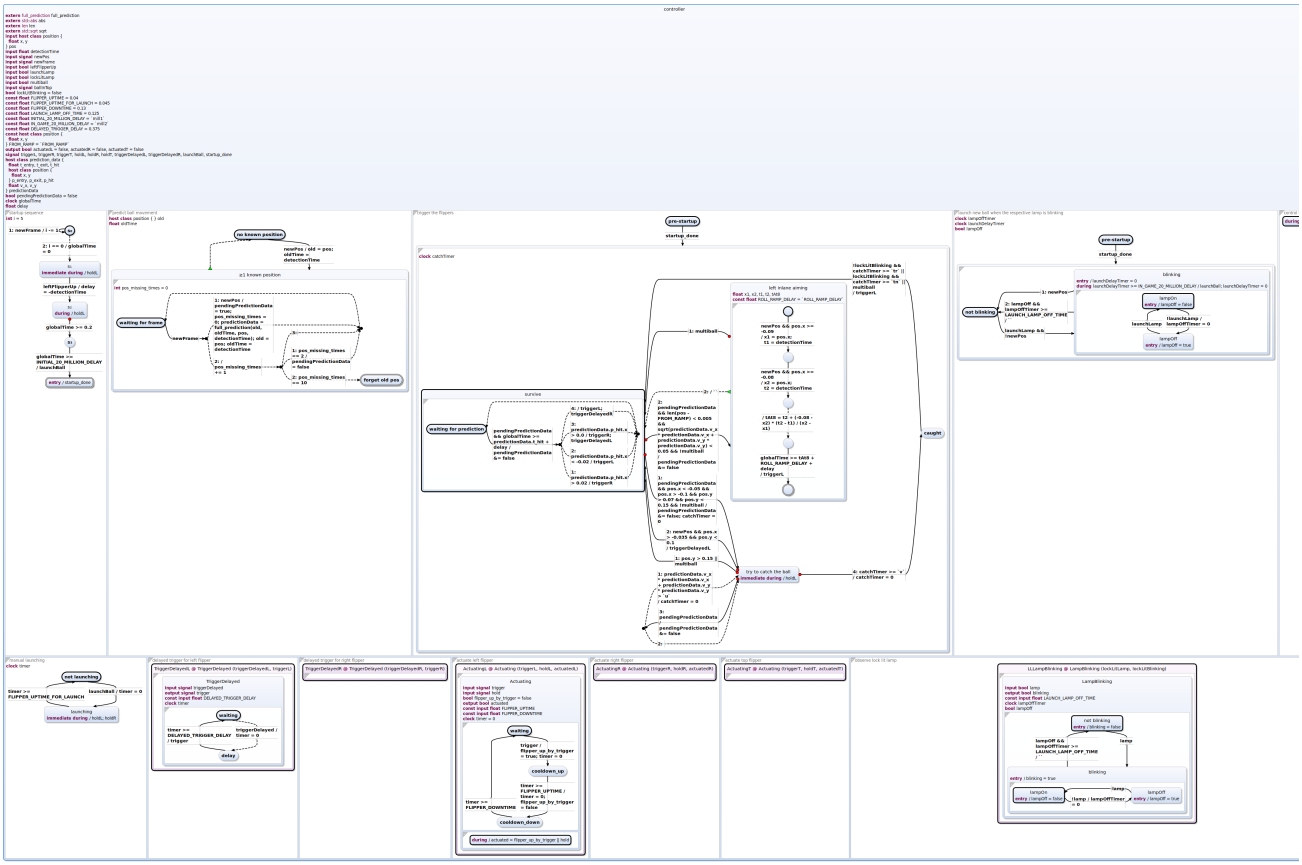
Key Functions

- Automatic launch when startup lamp is blinking
- Acquire 20 million point launch bonus
- Retrieve position predictions and actuate main flippers accordingly
- Catch ball on the left flipper and aim for some targets
- Actuate the top flipper if ball is visible around it

Overview



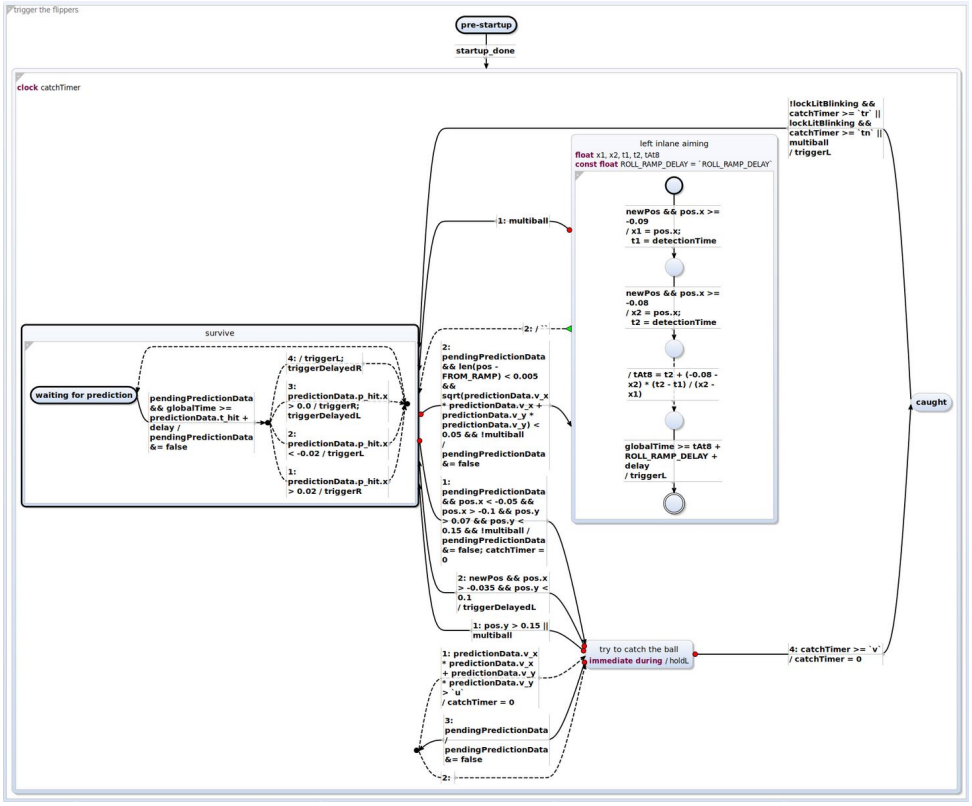




The behaviour and information processing is distributed across multiple regions in the controller.

- Yellow: These regions are in control of setting the output signals. These output signals are what is used to set the GPIO pins outside of the controller. The
  `delayed` regions are just a simple extension on top of the `actuate` regions.
- Red: These two regions control launching. `manual launching` just holds both flippers up for a long enough period for the machine to launch the ball into the game. The other region is in charge of detecting when the launch lamp is blinking and then waiting the appropriate amount of time to acquire the 20 million launch bonus.
- Blue: These are the main brain of the controller. These regions retrieve ball positions, ask for predictions on future ball positions, process a little game information to determine possible targets and decide when and how to actuate the flippers.

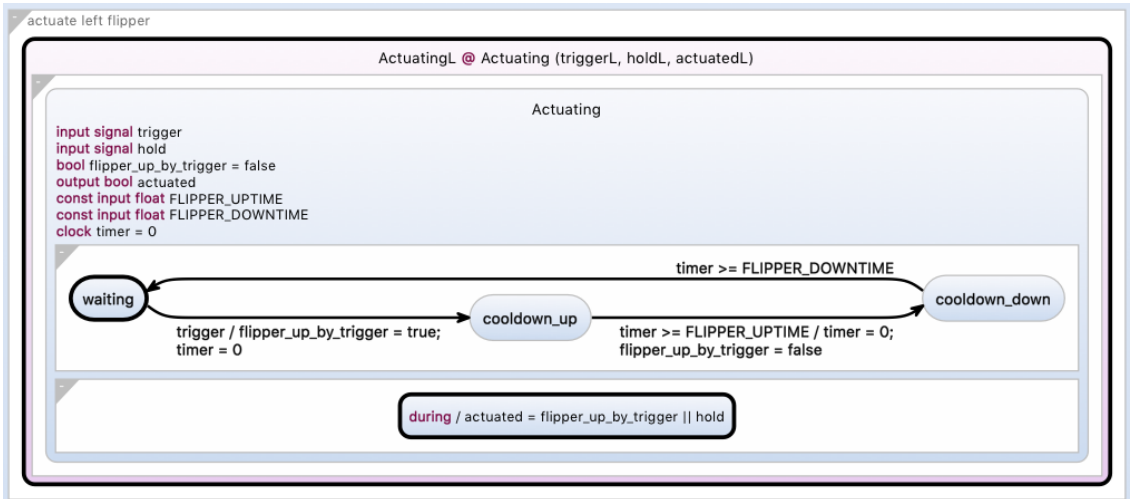
Main Flipper Controller



Arguably the most central element of a flipper autopilot is the module that decides when to trigger the flippers. In this system said decision is mainly made in the region displayed above. Most of the time while playing the automaton will be in the *survive* state, where it reacts to the times as given by the prediction. Here the only objective is to keep the ball in the game.

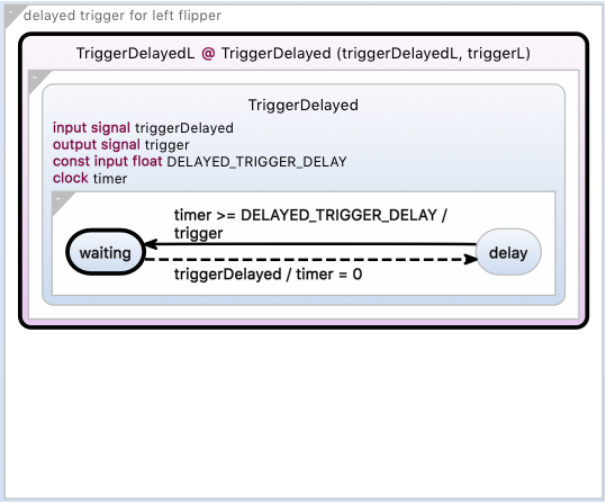
In certain situations this state is left in order to perform more elaborate gameplay maneuvers.
One of these is to catch the ball with the left flipper. Once this was successfully performed targeted shots to certain game elements are possible and the controller decides which one to aim for by analyzing some lamps that indicate certain game situations. The rule of thumb is: When there is a lamp lit above some target, it should be aimed at.
Another one is to detect a ball coming back after successfully hitting the big ramp. The ball is then led down a deterministic path feeding it into the left inlane. In this situation the controller tries to trigger the left flipper at the right timing to shoot the ball right into the big ramp again. This is useful since successive ramp shots are beneficial in most situations, as they score good points and lead to activation of a multiball mode.

Actuating



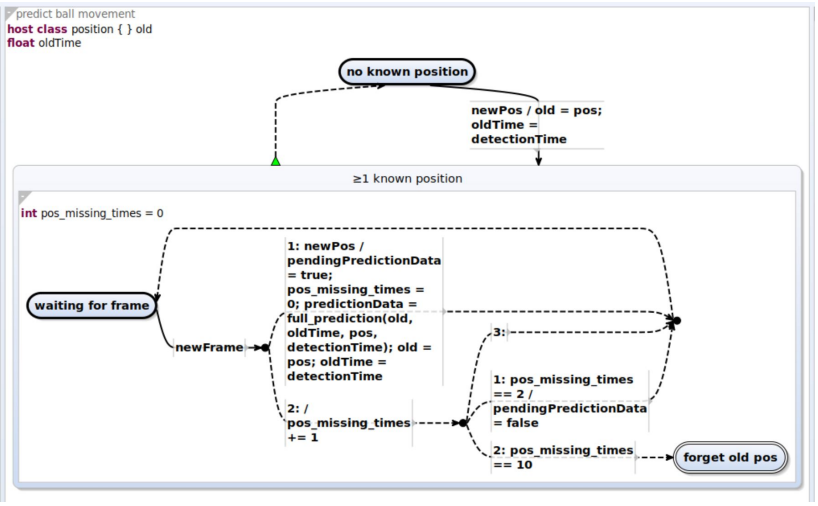
The actuation of flippers is implemented via a common automaton to which the appropriate signals are connected. The connected output signals are what is used to set the values of the GPIO pins by the outside system. With the `trigger` signal a standard actuation of the flipper can be initiated. The automaton waits after enabling and disabling for the real world to catch up, as the flipper arms take some time to move from their resting position and back. Additionally there is a hold signal which can be used from the outside to basically override the behaviour regarding the cooldown.

TriggerDelayed



As previously mentioned these regions are simply an extension on top of the `Actuating` regions. They receive a trigger signal and then wait for a certain amount of time until passing that information onwards (ie. setting the trigger signal connected to one of the `Actuating` regions seen above.

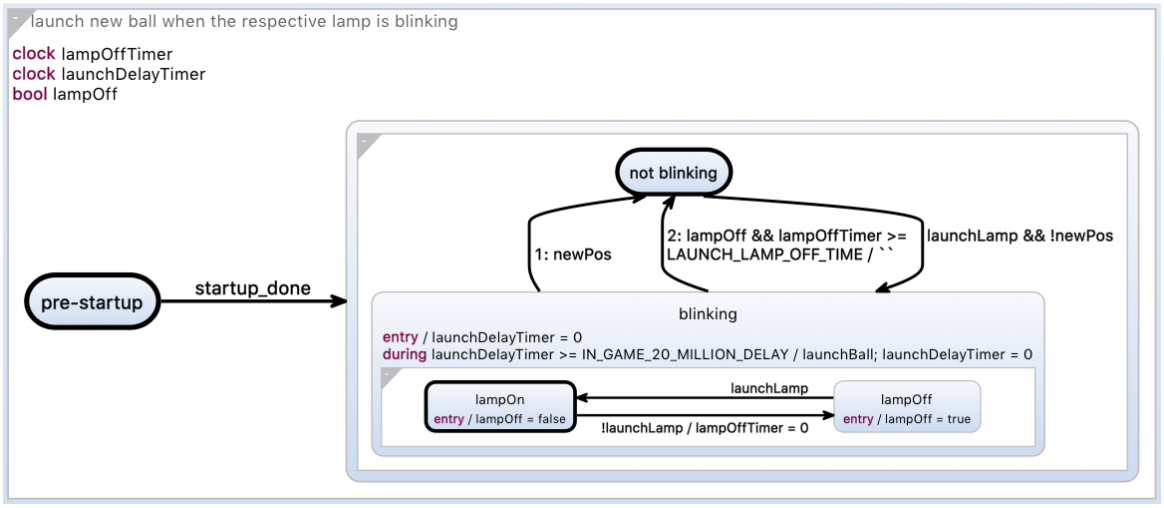
Prediction



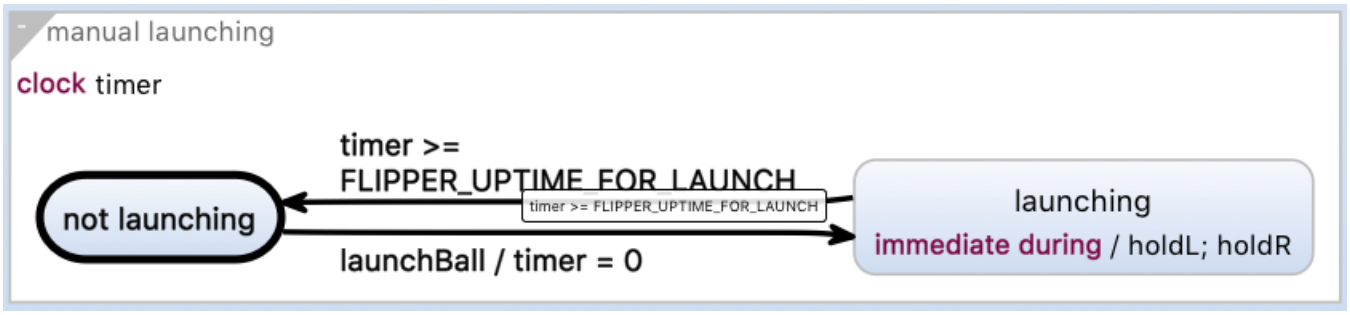
This region reacts to the input signals *newPos* and *newFrame*, that are coming from the image processing.
Whenever there are two temporally close positions, where the ball was detected, an external function call to the ball physics model is made.

When the ball was not detected for several frames, e.g. when it fell to the drain, or left the detection's region of interest, the information about the last position becomes stale and should be discarded.
In order to decide when to discard an old position the automaton counts frames since the last recognized ball position. After two frames the position is marked as no longer pending and after 10 frames it is discarded.

Launching



This first region determines when the launch lamp is blinking (ie. a ball is ready to launch) and when to launch. When the lamp turns on and the automaton has not received a new ball position from the outside system we assume it is starting to blink. We keep track of the state of the lamp and if for example it only turned on once a cooldown timer will run out and we determine it is not continuously blinking like it does when we are able to launch a ball into the game. If the automaton determines the lamp to be blinking for a long enough time (the 20 million points bonus timing), then a signal is set which is consumed by the following region.



Here we simply set signals that actuate both flippers for a time period so that the pinball machine launches the ball. If this time is too short both flippers will still actuate, but the pinball machine won't launch the ball.

Control Top Flipper

This region simply triggers the top flipper whenever there is a ball detected to be close to its range. Hitting the ball with the top flipper is often beneficial but never critical. So in order to save resources there is less computational effort put into it, i.e. no prediction is performed and the range is not precisely defined. It is implemented as a single state with a during action, since the work of detecting, whether there is a ball near the top flipper, is performed by the image processing, which sets a flag in the respective case.