**Synchronisatie** **mechanismen**:

**RunGameTaak:**

* **rtos::channel<buttonid, 10> inputChannel**

Gekozen voor een channel aangezien elke waarde van belang is tijdens het spel en voorgaande waarde dus ook gebruikt moeten worden. De channel is 10 groot. Meer dan 10 button presses achter elkaar zullen nooit gedaan worden, dus ook niet nodig deze hoger dan 10 te hebben.

* **Rtos::flag messageFlag & rtos::pool<uint32\_t> messagepool**

Gebruik gemaakt van een combinatie van flag en pool. Hiervoor is gekozen omdat alleen de laatste waarde belangrijk is. Als je wordt geraakt of het startsignaal wordt gegeven is er namelijk een time out voordat je weer geraakt kan worden. Daarom zou het overbodig zijn om elke keer de channel te clearen.

* **rtos::clock secondClock**

Gekozen voor een clock zodat elke seconden door wordt gegeven dat er weer een seconde voorbij is van de gametime en dergelijke.

* **rtos::timer delaytimer**

Gekozen voor een timer aangezien deze voor meerdere waits wordt gebruikt. De timer wacht waarde kan verandert worden en is daarom geschikt voor de verschillende wacht tijden.

**SendTask**:

**rtos::flag SendFlag**

**rtos::pool<uint32\_t> sendpool**

Hier is gekozen voor de combinatie van een pool en een flag aangezien alleen de laatste waarde verstuurd moet worden. Het zou niet uitmaken of voorgaande waardes worden overschreven.

**SpeakerTaak**:

**rtos::flag ShootFlag**

**rtos::flag HitFlag**

Gekozen voor flags omdat er geen waarde doorgegeven hoeft te worden. Het moet alleen duidelijk gemaakt worden of je aan het schieten bent of dat je geraakt wordt.

**rtos::timer SpeakerTimer**

Gebruik gemaakt van een timer om er voor te zorgen dat de speaker de gewenste pieptoon produceert.

**KeypadTaak**:

**rtos::clock KeyClock**

Gebruikt om om de 100 ms de keypad te pollen om te kijken of er een knop is ingedrukt.

**InputControlTaak**:

**rtos::channel<char, 18> InputChannel**

Voor gekozen om een channel te gebruiken zodat er geen voorgaande input verloren gaat als er te snel wordt gedrukt. De grote 18 is ruim genomen zodat deze niet vol zal raken.

**DisplayTaak**:

**rtos::channel<TypeMessage, 10> inputChannel**

Gekozen voor een channel omdat er overlappende berichten binnen kunnen komen voor de verschillende indelingen van het scherm. Het eerste bericht kan bijvoorbeeld de timer veranderen rechtsonderin terwijl het 2e bericht een tekst op het midden van het scherm wil laten zien. Deze willen we natuurlijk beide printen en niet dat voorgaande berichten verloren gaan in een pool.

**ButtonTaak**:

**rtos::clock ButtonClock**

Gebruikt om om de 100 ms de button te pollen om te kijken of er een verandering in de waarde is.

**Pause\_detector**:

**rtos::clock interval\_clock**

100 us clock gekozen omdat er dan om de 100 us word gecheckt of de ir-receiver van output verandert.

**Msg\_decoder**:

**rtos::channel<int, 128> pauseChannel**

Gekozen voor een channel aangezien er geen waarde verloren mogen gaan en elke pauze belangrijk is. Er is gekozen voor een grote van 128 omdat er voor gezorgd moet worden dat de channel niet vol raakt en een bericht al 33 pauzes bevat.

**rtos::timer reset\_timer**

De reset timer zorgt ervoor dat als er een te lange pauze is de msg wordt gereset.