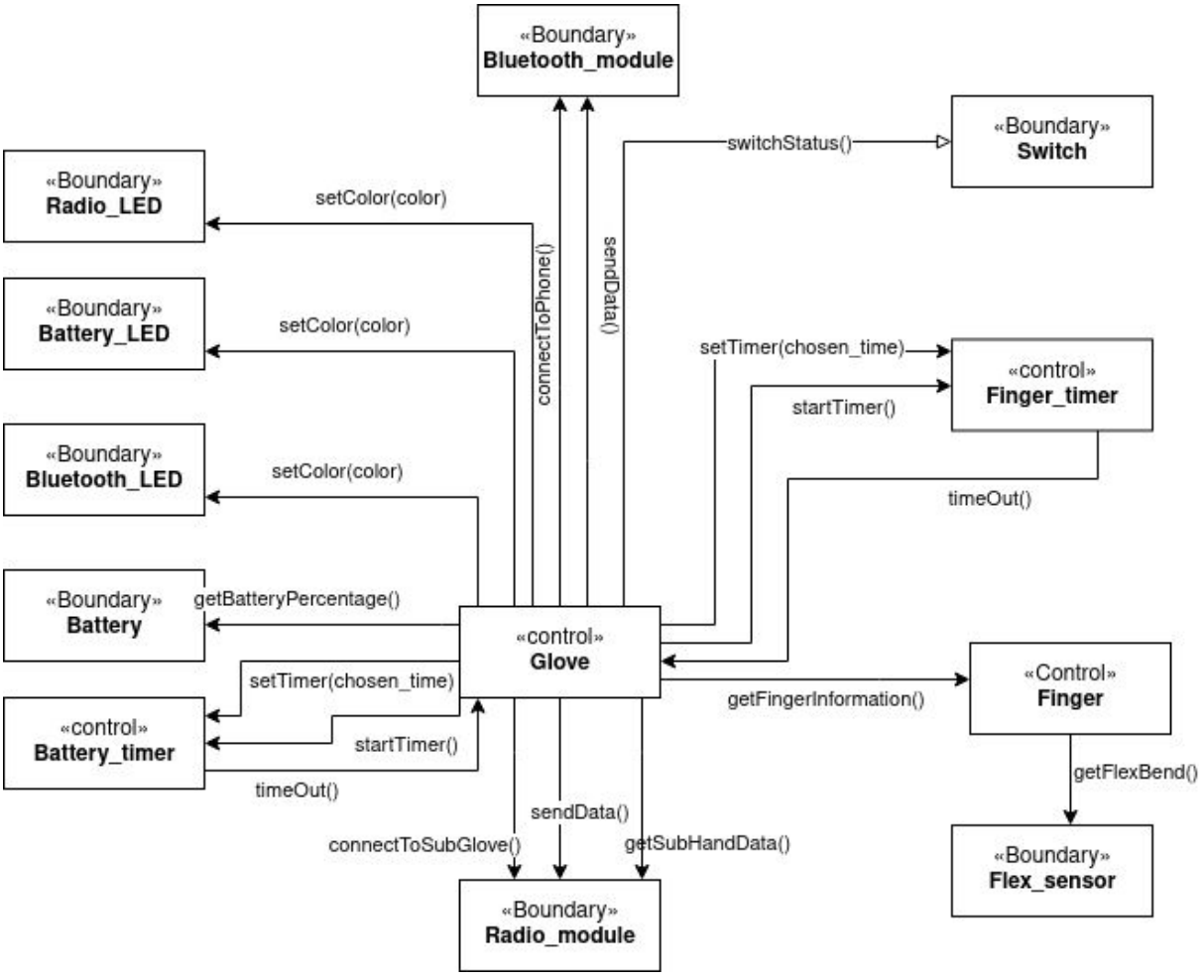
Modellen Honest Gloves

Object model

Binnen een object model laten wij zien uit welke bouwstenen de honest gloves bestaan. Dit is de eerste stap in het maken van een goed systeem.



Taakstructurering

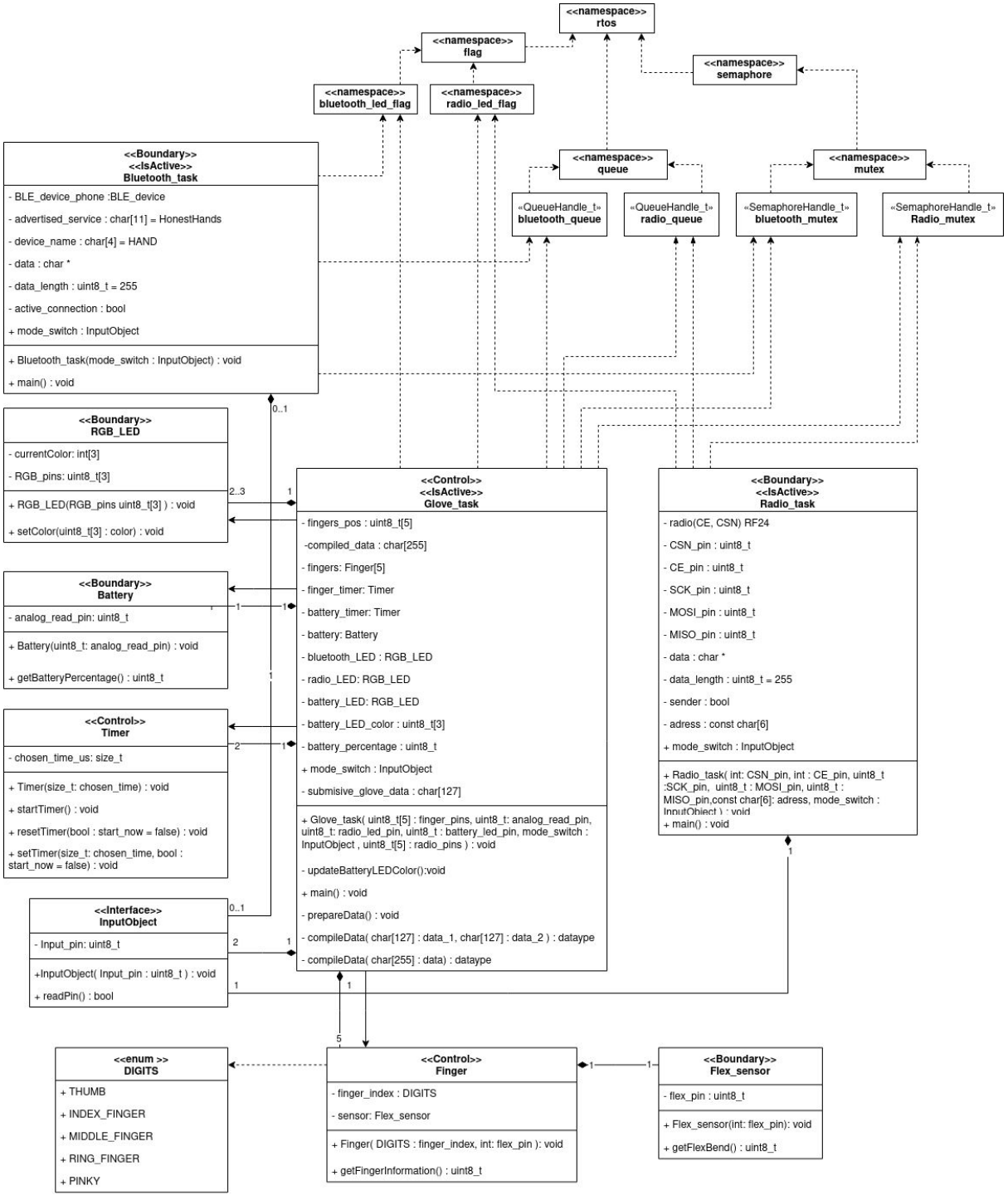
Hieronder weergegeven is de taakstructurering voor de honest glove handschoenen. Hierin is zichtbaar wat de prioriteit van verschillende taken zijn en welke objecten onder welke taak vallen. We hebben binnen dit systeem 3 afzonderlijke taken. Een glove task om de handschoenen mee te besturen en twee losse zend taken. Deze zend taken zijn voor de bluetooth en radio connectie.

Taak structuur				
Objecten	**Taaksoort**	**Periode**	**Deadline**	**Prioriteit**
Glove	intern, Periodical	30 ms	30 ms	1
Finger	intern, demand driven			3
Flexsensor	IO, demand driven			4
Connection LED	IO, demand driven			4
Battery LED	IO, demand driven			4
Button	IO, demand driven			4
Radio module	IO, periodical	50 ms	50 ms	3
Battery	IO, demand driven			5
Bluetooth module	intern, periodical	40 ms	40 ms	2
Finger Timer	-	-	-	
Battery Timer	-	-	-	

Taak clustering					
Task	**Objecten**	**Taaksoort**	**Periode**	**Deadline**	**Prioriteit**
Glove task	Finger, flex sensor, Glove, battery. Connection LED, battery LED	intern, demand driven	50 ms	50 ms	2
Radio task	Radio	IO, periodical	50 ms	50 ms	3
Bluetooth task	Bluetooth module	Intern, periodical	40 ms	40 ms	1

Class Diagram

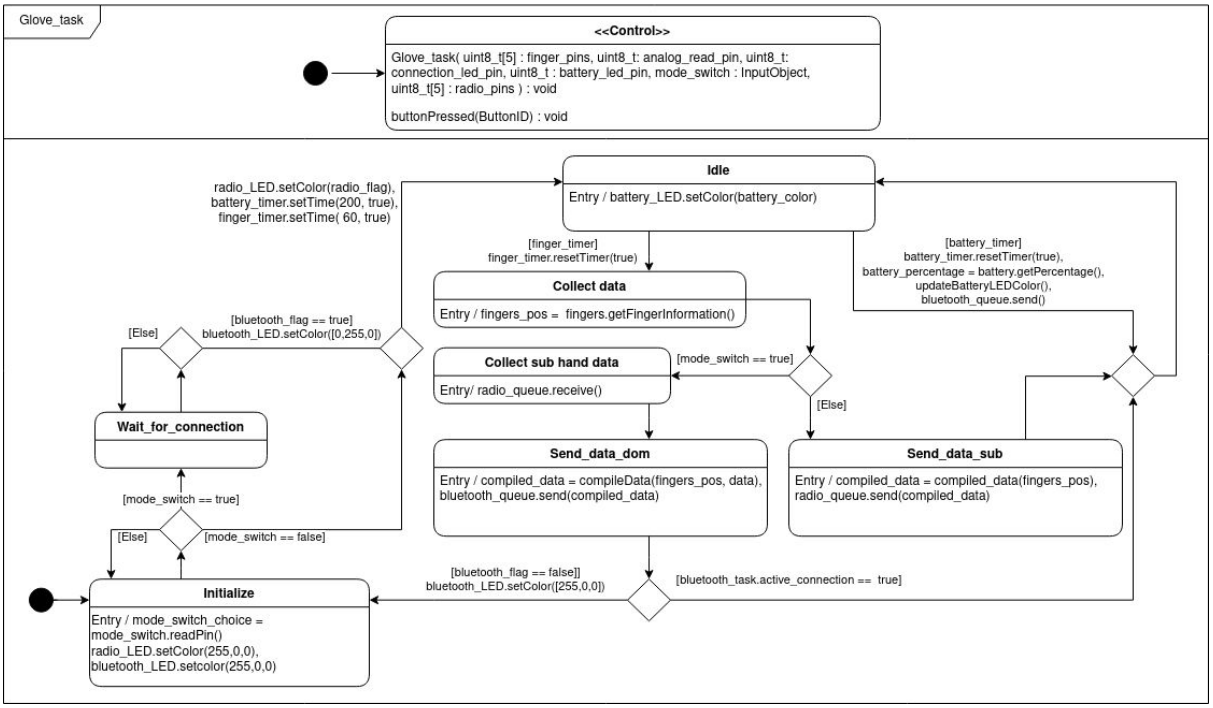
Binnen het klassendiagram wordt de structuur van de code uitgezet. Elke klasse heeft controle over een object of wordt gebruikt om data in op te slaan. Binnen ons systeem gebruiken wij ook namespaces, Aangegeven rechtsboven in het diagram. De grootste klasse is de glove. In de glove wordt het meeste gedaan en deze glove bezit dus ook veel andere klassen. Een glove “bezit” een batterij en een radio module enz. De glove bezit geen radio of bluetooth module aangezien dit losse taken zijn.



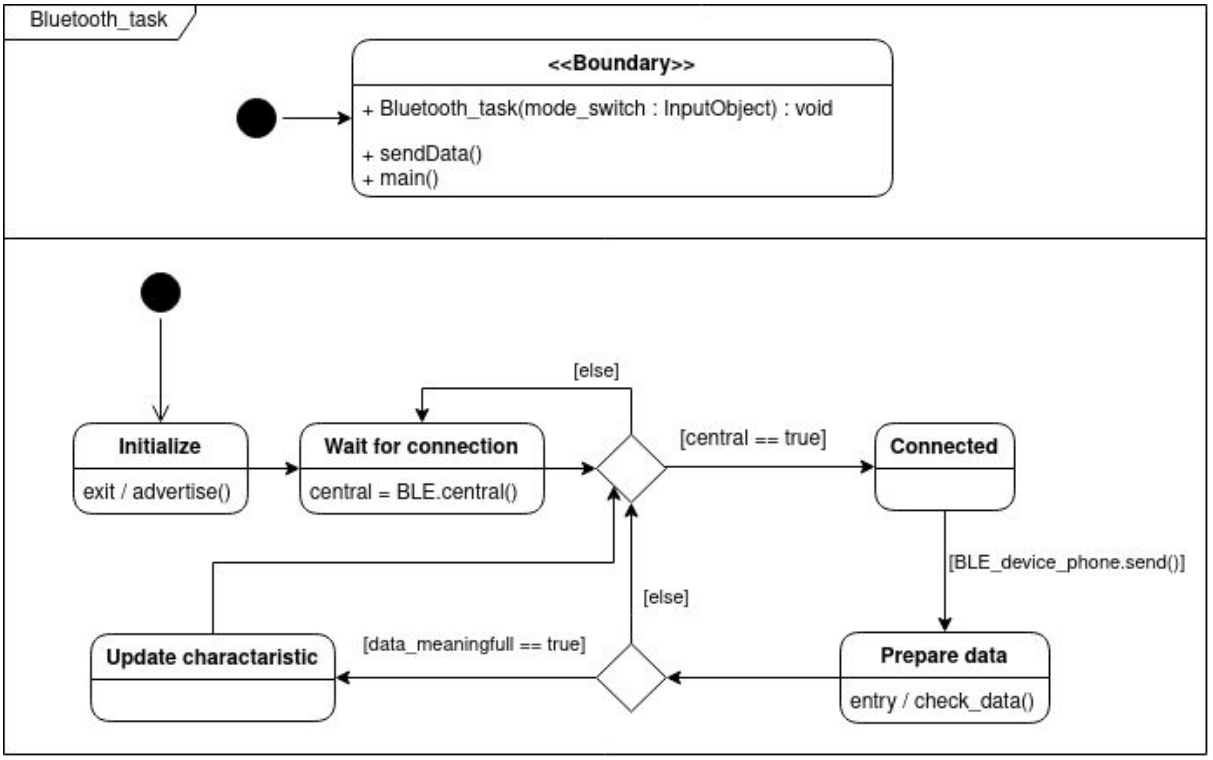
State transition diagrams

Binnen een state transition diagram wordt in diagramvorm het grootste deel van de code al uitgedacht. De diagrammen laat het proces binnen elke taak zien. Er is dus precies te volgen welke stappen de task doorloopt.

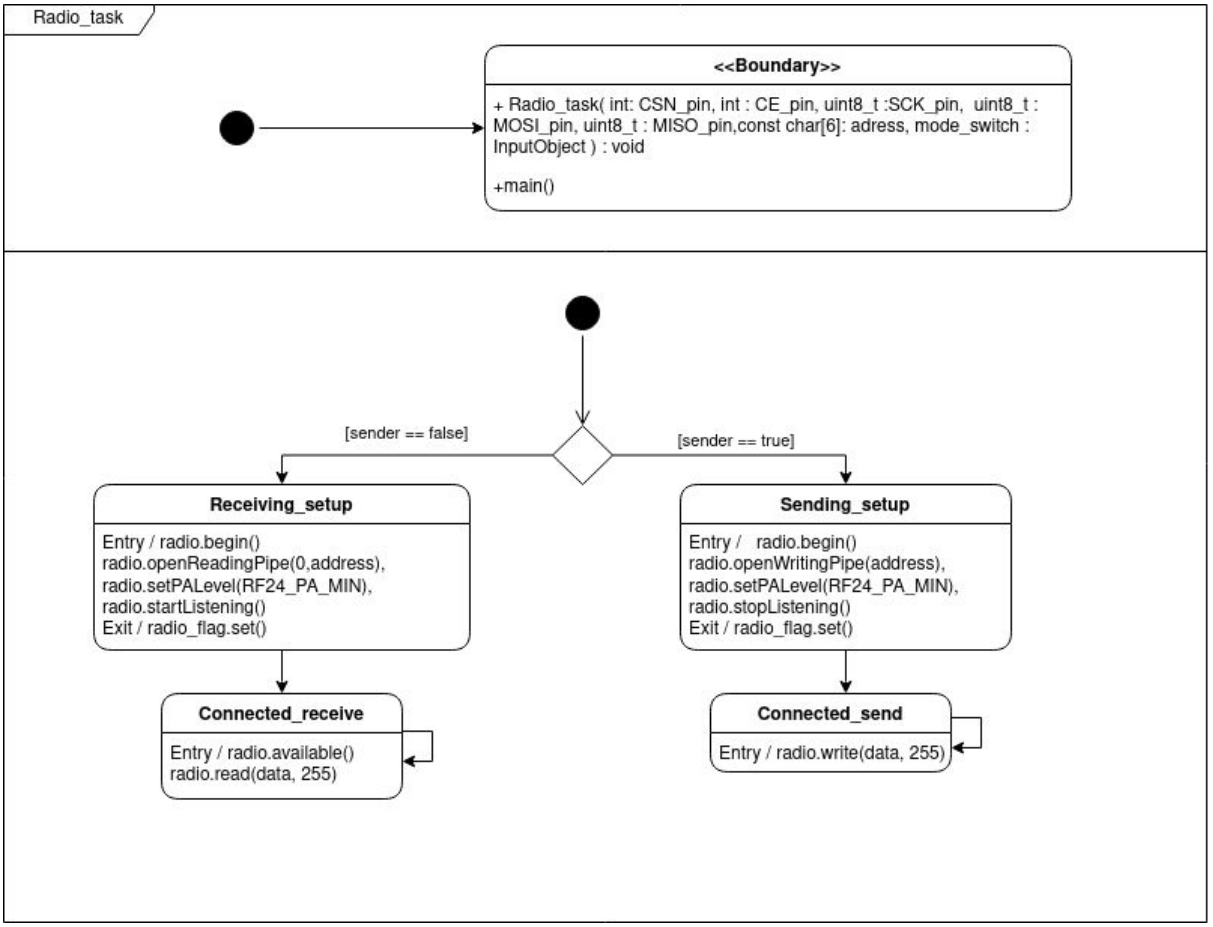
Glove task



Bluetooth Task



Radio Task


Radio_task
<<Boundary>>
+ Radio_task(int: CSN_pin, int : CE_pin, uint8_t :SCK_pin, uint8_t : MOSI_pin, uint8_t : MISO_pin,const char[6]: adress, mode_switch : InputObject) : void
+main()
[sender == false]
[sender == true]
Receiving_setup
Entry / radio.begin()
radio.openReadingPipe(0,address),
radio.setPALevel(RF24_PA_MIN),
radio.startListening()
Exit / radio_flag.set()
Sending_setup
Entry / radio.begin()
radio.openWritingPipe(address),
radio.setPALevel(RF24_PA_MIN),
radio.stopListening()
Exit / radio_flag.set()
Connected_receive
Entry / radio.available()
radio.read(data, 255)
Connected_send
Entry / radio.write(data, 255)