Voorverhaal

Aan het begin van het project stond ik voor twee mogelijkheden. Ik wist namelijk dat ik iets wilde doen met Arduino (of NodeMcu om precies te zijn, dat is een sterkere Arduino met Wifi). De ene mogelijkheid was kijken of ik code van de Arduino kon runnen op de computer. Aangezien Arduino code gebaseerd is op C++ en je ook C++ kan gebruiken leek dit best te doen. Op de achtergrond is het namelijk zo dat de Arduino IDE de code compiled met een C++ compiler naar code die gerund kan worden op Arduino. Ik zou dan een eigen compiler moeten bouwen die de Arduino code kon compilen naar een .exe, zoals dat gaat met normale C++ programma's. Na enig onderzoek bleek echter dat zelf compilen kon, met een "makefile" in Linux, hier hield het voor mij op.

De tweede mogelijkheid was om mijn Arduino te gebruiken als een controller. Ik had al gewerkt met mqtt, dus dat zou beter moeten lukken. Deze aanpak is ook uiteindelijk gelukt. Een probleem was wel dat, zoals te lezen is in 'strengths & weaknesses' ik mijn telefoon moest gebruiken als hotspot en hierdoor ging ik door mijn bundel heen. Uiteindelijk ben ik uitgekomen op een http versie waarbij mijn Arduino een soort website hostte, als een soort API. Ik heb wel besloten om beide versies in mijn project te houden omdat ik dan op twee plekken mijn code kon splitsen (hetgeen wat nodig was voor dit vak). De mqtt oplossing gebruikte een Asset, omdat zelf NuGet importeren in Unity een hele klus was, hierdoor zitten er zo'n 20 extra scripts in het project. De http oplossing gebruikt standaard C# functionaliteit.

Met dit soort grote projecten merk je wel wat de limitaties zijn van Unity, vooral omdat het een verouderde versie van C# gebruikt. Veel "mooie" oplossingen kunnen niet en sommige dingen kunnen überhaupt niet.

Strengths & weaknesses

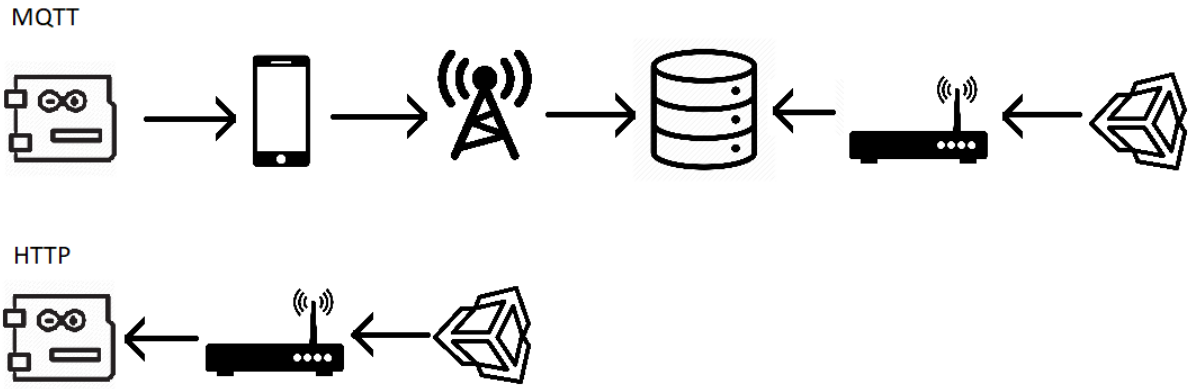
De eerste splitsing zit in compilervariabelen. Dit is de keuze tussen de eigen input van Unity en de ArduinoInputs. Aangezien de ArduinoInputs altijd afhankelijk zijn van een form van internet of een netwerk, kan het dus zijn dat Arduinoinputs niet werkt. Doordat de Unity input er niet is op dat moment, kan de game niet gespeeld worden. Er zou nog een Usb Arduino implementatie kunnen worden toegevoegd in het geval dat geen van de andere opties werken, maar dit heb ik (nog) niet gedaan.

MQTT vs HTTP

De tweede splitsing zit hem tussen mqtt en http. Beide hebben hun voor en nadelen. Aan de ene kant is mqtt heel handig omdat je daarmee vanuit de Arduino informatie kan sturen, in plaats van informatie uitlezen constant. Ook kan het in mijn geval over het hele internet, dat kan niet met mijn http implementatie. Dit heeft te maken met de poorten die open en dichtstaan op SSH internet en Eduroam. Bij mqtt is de controller ook gescheiden van Unity, dus als een van de twee wegvalt blijft alles nog werken, maar er zitten ook nadelen aan. Mqtt maakt gebruik van een extra server, waardoor alle informatie meer afstand moet afleggen. Eerst gaat het van Arduino via een hotspot van de telefoon naar de mqtt server, om vervolgens van de server naar Unity te gaan. Deze afstand vertaalt zich in input delay. Ook is het in mijn geval onhandig dat ik constant mijn telefoon als hotspot moet gebruiken. Ik zou dit kunnen omzeilen door een mqtt server op mijn computer te installeren en gebruik te maken van een lokaal netwerk, maar dat is me niet gelukt.

Aan de andere kant is er de http implementatie. Hierbij verbind ik de computer en de Arduino met een router om een lokaal netwerk te maken. Unity doet webrequests naar de Arduino en op dat

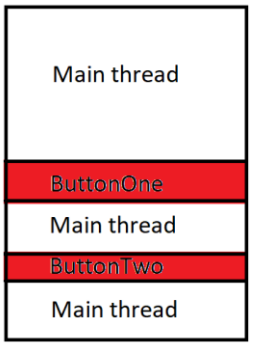
moment geeft de Arduino zijn waarde terug. Een ping op deze manier is 4ms, dat is ontzettend snel. (op de mqtt manier kon ik geen ping doen, maar ik schat het op zo'n 500ms). De http manier is dus een stuk sneller en makkelijker, maar ik kan het niet combineren met internet (dit komt weer door SSH en Eduroam). Dus als internet niet nodig is, dan is dit in mijn ogen de beste oplossing.



Informatiepaden van de verschillende opties

Button debounce

Beide implementaties zitten met het probleem van de button inputs opslaan. Unity of Arduino moet namelijk niet alleen weten wanneer een button is ingedrukt, maar ook wanneer hij dat weer niet is. Ik heb daarom gekozen voor een oplossing waarbij Arduino constant stuurt wat de staat is van de buttons bij mqtt en dat http constant vraagt wat de staat is. Het probleem hiermee is wel dat er alleen gecheckt wordt op bepaalde momenten. Bij mqtt is de buttonState afhankelijk van de frequentie van checken van de Arduino. Bij http is het zelfs erger, maar daarvoor is een beetje uitleg nodig. http werkt met co routines in Unity, maar unity werkt niet met meerdere threads dus co routines worden samen met de main thread uitgevoerd in één thread. Dat betekent dat, hoe meer co routines, hoe langer het duurt voordat elke co routine wordt uitgevoerd. Hoe complexer het spel wordt en hoe meer co routines er komen (voor elke knop is er één co routine), hoe minder vaak er kan worden gecheckt of een knop is ingedrukt. En hoe minder vaak er gecheckt wordt, hoe kleiner de kans is dat er gecheckt wordt op het moment dat een knop is ingedrukt, ergo, hoe langer een knop ingedrukt moet worden om geregistreerd te kunnen worden. (Onderaan staat een linkje over co routines in Unity)



Door timeslicing kun je niet zeggen wanneer een button geupdated gaat worden

Vanuit de mqtt kan dit opgelost worden door de buttons te debouncen in rduino. De http kant is alleen wat moeilijker, omdat het ook afhankelijk is van Unity zelf en daar heeft Arduino niets aan natuurlijk.

Ik stop alle inputs die van de Arduino komen in een Dictionary. In mijn voorbeeld van 2 inputs is het sneller om een array te gebruiken voor dit natuurlijk, maar voor grotere projecten is het wel sneller om een Dictionary te gebruiken, daarom heb ik hem er in gezet als een dictionary.

Blocking vs Non-Blocking

Een ander belangrijk detail is hoe mijn WebManager werkt. Bij de WebManager gaat een foreach loop door alle IWebHandlers heen om te kijken of ze werken. Ik heb hierbij misbruik gemaakt van

een aantal functies. Ik maak namelijk gebruik van een WebRequest, dat is een functie die in een thread gebruikt moet worden omdat een blocking functie is. Ik heb er alleen voor gekozen om het in de main thread te doen, zodat het programma wacht. Dit is ten eerste om er voor te zorgen dat het programma pas verder gaat met zoeken als de huidige methode niet werkt (dit zou ook anders opgelost kunnen worden door naar alles te zoeken en van alle werkende resultaten er een te kiezen). De tweede en belangrijkste reden om dit te doen is dat Unity pas start zodra alle Start functies zijn uitgevoerd. Door deze blocking manier start Unity pas op het moment dat er één IWebHandler werkt, of als niets werkt. Een nettere uitwerking zou zijn om een laadscherm te tonen terwijl Unity opzoek is, maar dit project is te kleinschalig om die functionaliteit allemaal toe te voegen.

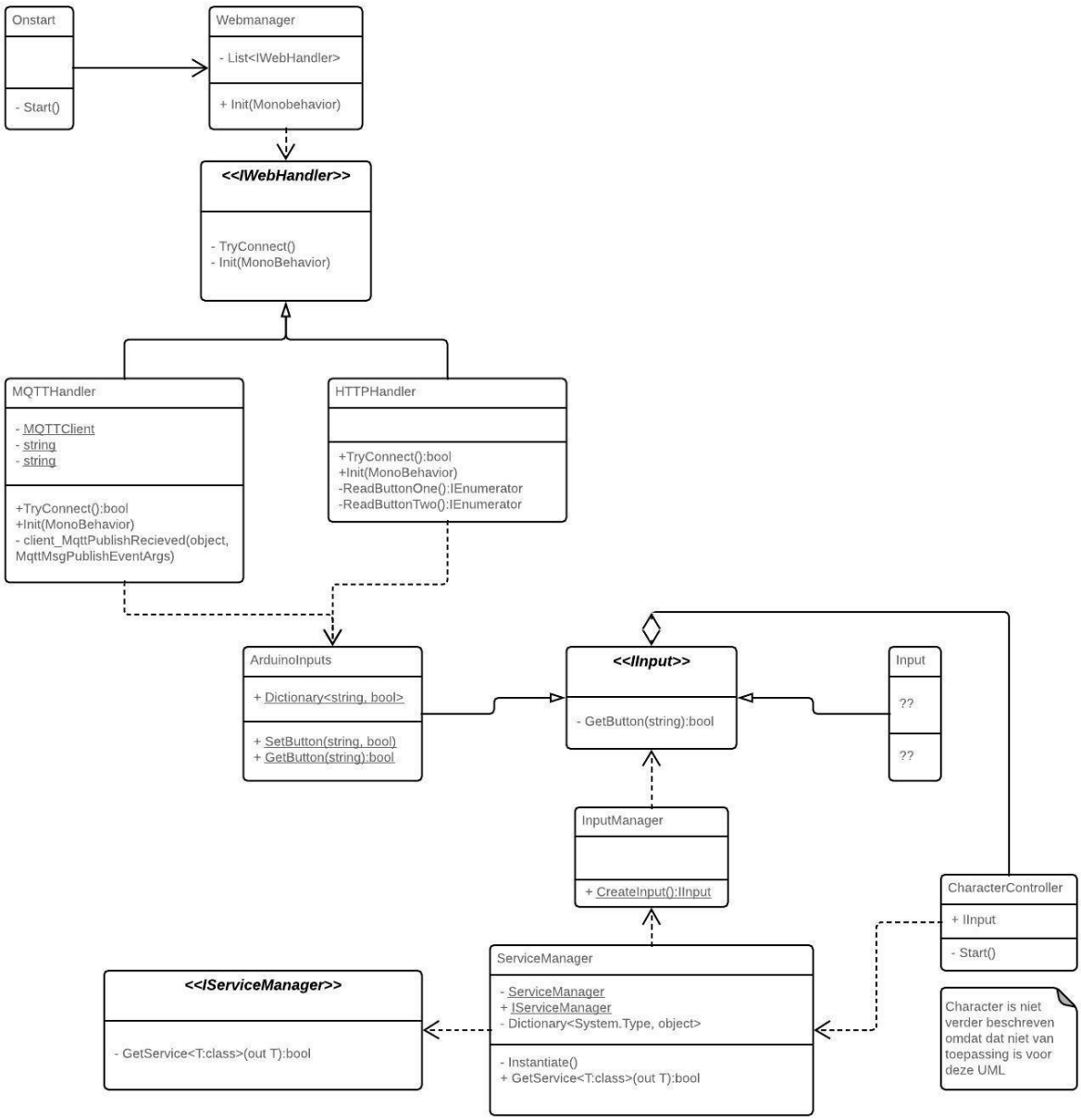
Generic Input

Bij de bespreking hebben we nog kort gekeken naar het generic maken van de inputs.

```
#region Unity Input

public class UnityInput : IInput
{
    public virtual bool GetButton(string key)
    {
        return Input.GetButton(key);
    }
}

#endregion

#region Arduino Input
public class ArduinoInput : IInput
{
    public virtual bool GetButton(string key)
    {
        return ArduinoInputs.GetButton(key);
    }
}
#endregion
```

Dit zou kunnen werken als ik niet de Unity input class zou gebruiken. De constraint die ik zou zetten op de type zou namelijk de IInput interface zijn, aangezien zowel de Input en ArduinoInputs classes een GetButton functie hebben. Ik kan alleen niet die interface zetten op Input omdat dat allemaal intern geregeld is door Unity. Hierdoor is het, voor zover uit mijn onderzoek bleek, niet mogelijk, maar ik hoor het graag als dit toch wel blijkt te kunnen. Ik heb dit voorbeeld verder uitgelegd in een bijgeleverd filmpje.

Uml



Links

https://support.unity3d.com/hc/en-us/articles/208707516-Why-should-I-use-Threads-instead-of-Coroutines-