Project Proposal

Kacper Dabrowski

21237298

# National College of Ireland

**HDip in Computing**

(**HDSDEV\_SEPBL\_YR2**)

### Distributed Systems Project

## OVERVIEW

*This document is submitted as an interim submission of CA project of the Distributed Systems module on the Higher Diploma in Computing specializing in Software Development delivered by Yasantha Samarawickrama.*

# Domain description

# *You should best describe the overall purpose of the service, explain the functionalities within each service and overall contribution of the service to the application.*

# *[200 words]*

**(As my student number ends with 8 my project will be smart farm (agriculture).)**

Smart farms are a rapidly growing market due to the latest advancements in remote robotics and AI fields, combined with a growth of remote/ cloud computing services. It goes way beyond smarts sensors and weather forecasts, and it starts to include the use of remote tractors, AI harvester robots (that can also detect weeds and spray pesticides) and remote drones to expand scope of smart possibilities in agriculture. It can use satellite imagery to map regions of farms (for example for remote tractors to operate) and AI for decision-making even regarding crops to be sown*. (Cropin, 2024) (Elsayed Said Mohamed, AA. Belal, Sameh Kotb Abd-Elmabod, Mohammed A El-Shirbeny, A. Gad, Mohamed B Zahran,, 2021)*

My smart farm project implementation/ simulation will consist of services for smart climate for a greenhouse (or rather for many greenhouses), for remote use of self-driving tractors, drones and harvester robots and for field soil irrigation. *Greenhouse* needs specific climate and CO2 levels and light to assure best environment for plant growth. GreenhouseService would take stream of measurements from sensors (temperature, humidity, light, soil, moisture, CO2 level, amount of light) and change climate settings to assure best environment for plants. RemoteVehicles service would allow robots to send their position and receive stream of directions of where to go or what to do (harvest fruit or take measurement etc). SoilIrrigation service gets stream of measurements of soil moisture and send information if watering is needed.

# Service definition and RPC

# *You should explain in detail, with example the request and response for each functionality within the service. Explain in detail the parameters*

# *[300 words]*

I assume that, for IoT devices I should deliver terminal clients, and GUI client is requested only for user application.

In farm.proto file I defined services:

**Greenhouse Service** that allows to rpc ‘s :

1. *registerGreenhouse* (server side streaming) where client sends *RegisterGreenhouse* message and receive stream *ServerRequests*
2. *getClimateSetting* (bidirectional streaming) where client continuously sends a stream of *SensorMeasurements* (temp, humidity, soil moisture, amount of light, CO2 level) and receives *ClimateSettings* if any reading is outside of desired level. Client sends readings in time intervals(for example 30 minutes).

**Soil Irrigation Service** that has 3 rpc’s:

1. *registerDevice* (server side streaming) that registers soil moisture sensor or sprinkler for area (*RegisterArea)*, and receive stream *ServerRequests* (server can use that to turn on sprinklers when needed)
2. *sensorReading* (client streaming) where sensor send a stream of measurements (*SoilConditions)* in time intervals and receive *ServerMessage*
3. *turnOnOffWater* (unary) where sprinkler send information that it is tuned on (*SprinklerOn*) and waits for server to send *TurnOffReply* message to shut down water.

**RemoteVehicleService** that has 6 rpc**:**

1. *registerDevice* (server stream) that sends *RegisterVehicle* message and returns stream of *ServerRequests.*
2. *getRoute* (server streaming) where vehicle sends it’s *Position* and gets a stream of *Position*’s that he must travel to get to new place
3. *getTask* (unary) where vehicle requests what task it has to perform in a give *Position,* server return *Task.* (it depend on vehicle like plow for tractor, harvest (tomatoes/apples etc) for harvest-robot, take sensor reading for drone)
4. *recordTaskFinished* (unary) vehicle sends *Task* that it finished to record it in database, server sends *TaskSaved* message.
5. *vehicleStatus* (client streaming) vehicle sends stream of *Status* messages that contains fuel level, amount of space left (in container for harvested plants) etc, server can send *GoHomeMessage* if any reading prompts it to.
6. *getDroneRoute* (bidirectional streaming) where drone as being airborne must send stream of *Position*s and receives stream of *PositionWithTask* to make continuous work without delays (so it gets Task before he gets to position)

**UserAppService** with 12 rpc’s (at the moment) that allows user app to get readings and send requests to all devices.

1. *registerUserApp* (unary) for registering user app on server, sends *RegisterApp* server returns *Registered* message
2. *getGreenhouses* (server side streaming) client sends *GreenhouseListRequest*  and gets stream of *Greenhouse*s
3. *getGreenhouseData* (server side streaming) client sends *Grenhouse* to get sensor readings from that greenhouse and server sends stream of *SensorMeasurements.*
4. *getGreenhouseClimate* (unary) client sends *Greenhouse* and gets set of climate settings (in one message)
5. *getSoilAreas* (server side streaming) client sends *AreaListRequest* and gets a list of *Areas* where sensors (and sprinklers) are registered.
6. *getSoildData* (unary) client request soil conditions for *Area,* and receives *SoilConditions* with sensor readings (of soil moisture).
7. *getSprinklerStatus* (unary) client sends *Area* message and gets *SprinklerStatus* (on or off) for that area.
8. *getAllSprinklersStatus* (server side streaming) client sends *SprinklersStatusRequest*  and server sends stream of *SprinklerStatus* (on/off)
9. *turnOnOffSprinkler* (unary) client sends *Area* to to change sprinkler’s status in that area (from on to off or from off to on) server returns *SprinklerStatus*
10. *getVehicles* (server side stream) client sens *VehicleListRequest* and revceive stream of *Vehicle.*
11. *setVehicleTask* (server side stream) client sends *VehicleTask*  with vehicle,task and route for vehicle, server return stream of *VehicleStatus* so user can see progress of task
12. *getRecordedTasks* (server side stream) client sends *Vehicle* message and receives stream of *Tasks*that vehicle finished.

# Bibliography

Cropin. (2024, Nov). *Cropin*. Retrieved from https://www.cropin.com/smart-farming

Elsayed Said Mohamed, AA. Belal, Sameh Kotb Abd-Elmabod, Mohammed A El-Shirbeny, A. Gad, Mohamed B Zahran,. (2021). *Smart farming for improving agricultural management*. Retrieved from Science Direct: https://www.sciencedirect.com/science/article/pii/S1110982321000582