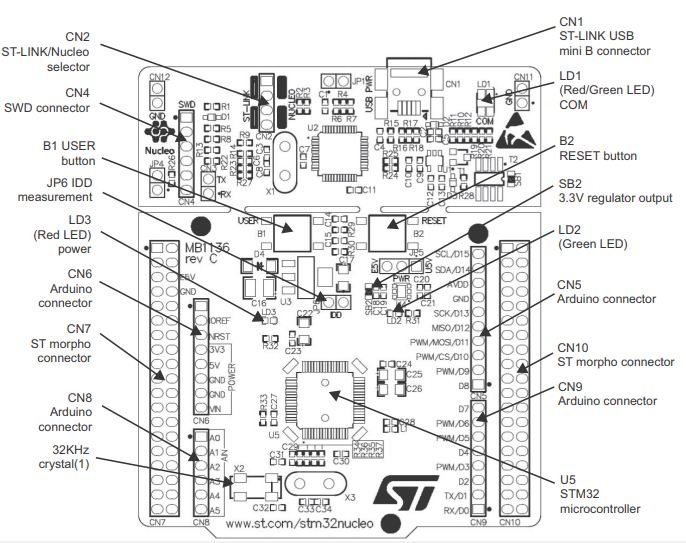
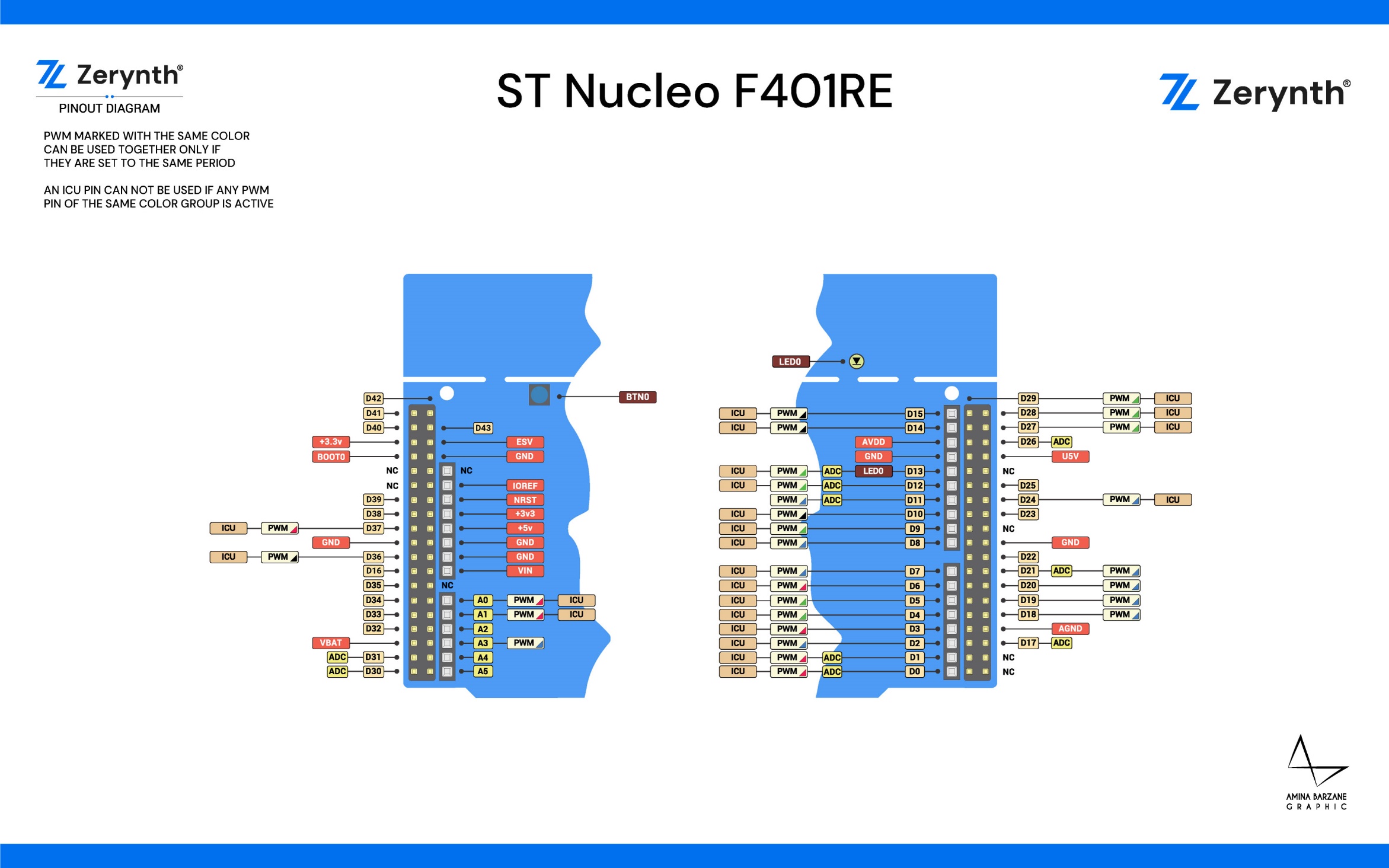
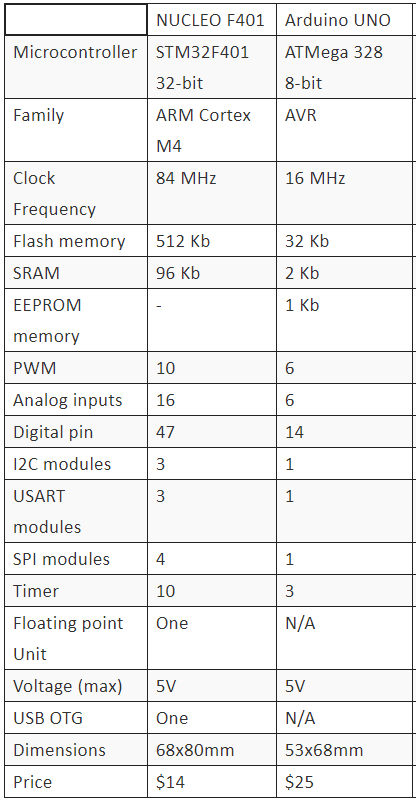
**STM NUCLEO F401re**

The STM NUCLEO F401re is a microcontroller board similar to the Arduino UNO but has many more ports and is used in many more places. It is a 32-bit board with ARM Cortex M4 CPU with FPU. It consumes 2.4uA at standby without RTC. The MCU has an operating voltage of about 1.7V to 3.5V whereas the board operates at arrange of between 7 to 15V. It can store up to 512kb in its flash memory and 96KB in its static RAM.



|  |  |  |  |
| --- | --- | --- | --- |
| **Pin Category** | **Pin Type** | **Pin names** | **Description** |
| CN7 | Port pins | PC0, PC1, PC2, PC3, PC10, PC11, PC12, PC13, PC14, PC15 | Port C digital I/O pins |
| PD2 | Port D I/O pin |  |  |
| PA0, PA1, PA4, PA13, PA14, PA15 | Port A I/O pins |  |  |
| PB7, PB8 and PB9 | Port B I/O pin |  |  |
| PH0 and PH1 | Port H I/O pins |  |  |
| Power | VBAT | Can be used to power them module from battery |  |
| +3.3V | Provides 3.3V as output can also be used to power the MCU |  |  |
| +5V | 5V output only pin |  |  |
| VIN | Unregulated input power pin |  |  |
| RESET | Resets the MCU |  |  |
| IOREF | Reference Voltage Pin |  |  |
| CN10 | Port Pins | PC4, PC5, PC6, PC7, PC8, PC9 | Port C I/O Pins |
| PA2, PA3, PA4, PA6, PA7, PA10, PA11 and PA12 | Port A I/O Pins |  |  |
| PB1, PB2, PB3, PB4, PB5, PB6, PB8, PB9, PB10, PB12, PB14, PB15 | Port B I/O Pins |  |  |
| Power | U5V | 5V Power Pin |  |
| GND | System Ground of the MCU |  |  |

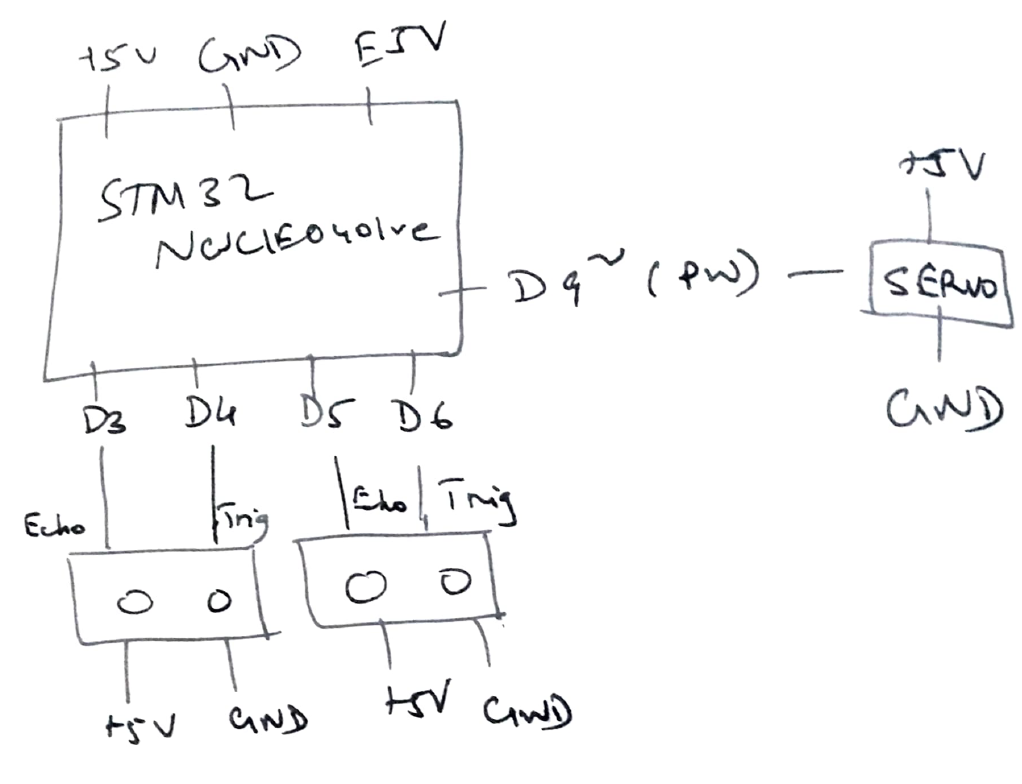




As seen in the above chart comparison of Arduino UNO and STM NUCLEO, various benefits for each microcontroller board is clearly visible.

The NUCLEO board doesn’t come with an internal EEPROM memory or even and internal EEPROM within the STM32 to store permanent variables in case of a system reboot, while the Arduino has a built-in EEPROM with the Atmel microcontroller.

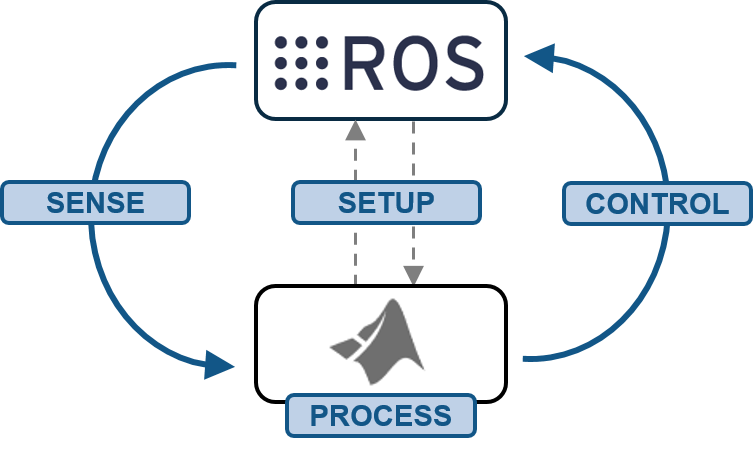
On the contrary the NUCLEO can process more complex algorithms due to the floating-point and also the compiler which is used is more effective as it will be written with far few instructions which allows quicker execution and significant efficiency within the programs. The STM NUCLEO boards are much cheaper which makes it a cost-effective alternative to the UNO. Also, for larger projects the STM would be a better buy as it has a greater number of digital pins making it easier for the connections.



**ROBOT OPERATION SYSTEM(ROS)**

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. The ROS is not a language but it is language independent which means that it is a programming language specification providing a common interface usable for defining semantics applicable toward arbitrary language bindings. The ROS helps us to create robots using these libraries rather than creating the programs from scratch which not only saves time but also the chances of a bug in the program.

ROS is not an OS but something called as a meta OS. To understand meta OS We need to understand what libraries and frameworks are before understanding Meta OS. Libraries are essentially groups of functions that are widely used in software/programs and are popular enough to justify packaging them into separate files. Libraries are also used to make the software look cleaner and to build upon the tried and tested software, leading to fewer chances of errors.



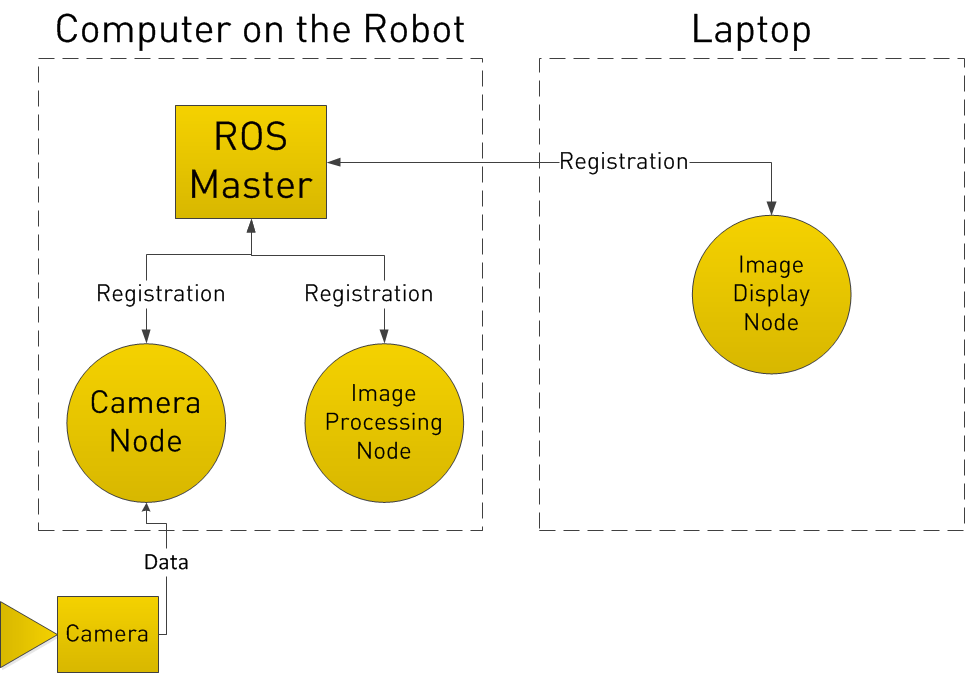
ROS demands a lot of functionality from the operating system. On top of that ROS must be freely available to a large population, otherwise a large population may not be able to access it. Much of the popularity of ROS is due to its open nature and easy availability to mass population. It also needs an operating system that is open source so the operating system and ROS can be modified as per the requirements of application.

Proprietary Operating Systems such as Windows 10 and Mac OS X may put certain limitations on how we can use them. This may lead to rigidity in the development process, which will not be ideal for an industry standard like ROS. Hence, most people prefer to run ROS on Linux particularly Debian and Ubuntu since ROS has a very good support with Debian based operating systems especially Ubuntu. That doesn’t mean that ROS can’t be run with Mac OS X or Windows 10 for that matter. But the support is limited and people may find themselves in tough situation with little help from the community.

Once the code is completed, we need to test our code so that we can make changes if necessary. Doing this on a real robot will be costly and may lead to wastage of time in setting up robot every time. Hence, we use robotic simulations for that. The most popular simulator to work with ROS is [Gazebo](http://gazebosim.org/). It has a good community support; it is open source and it is easier to deploy robots on it.

The robot will be setup with different sensors and actuators, luckily, we can find many of these in gazebo or otherwise build them on our own, which might take a long time but is still not very tough. While running these sensors, we may need to visualize their data. We use [RViz](http://wiki.ros.org/rviz) for this.

RViz is a 3D Visualization tool for ROS. It is one of the most popular tools for visualization. It takes in a topic as input and visualizes that based on the message type being published. It lets us see the environment from the perspective of the robot.



**MBED PROGRAMMING**

**a)**

int ledPins[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18};

void setup(){

for (int i=1;i<19;i++)

pinMode(i,OUTPUT);

Serial.begin(9600);

}

void loop(){

int n, t1 = 0, t2 = 1, nextTerm = 0,t,o;

Serial.println("Input a number");

n=Serial.read();

for (int j=0;j<n;++j){

nextTerm = t1 + t2;

t1 = t2;

t2 = nextTerm;

}

if (nextTerm<10){

t=0;

o=nextTerm;

}

else{

t=1;

0=nextTerm-10;

}

for(int k=0;k<t;k++)

digitalWrite((k+1),HIGH);

for(int m=0;m<o;m++)

digitalWrite((m+1),HIGH);

}

**b)**

#include <Servo.h>

Servo servo1;

Servo servo2;

void setup() {

servo1.attach(3);

servo2.attach(5);

Serial.begin(9600)

}

void loop(){

Serial.println("Enter value of angle 'a'");

a=Serial.read();

for (int i = 0; i < a; i++) {

servo1.write(i);

delay(10);

}

for (int i = 0; i < (180-a); i++) {

servo1.write(i);

delay(10);

}

}

**Automation in the Agricultural Sector**

The are many issues that can be tackled in the agricultural sector. The first and easiest issue is planting of seeds. A robot could be developed which would drop seeds at regular intervals of the ploughed fields. This would ensure uniformity in the fields as all plants would be in straight lines. The whole field would be geo mapped and this data could be fed to the bot. Using this data, the bot will be able to plant the seeds in a large area without any human supervision. The map would contain soil properties like pH water content and nutrients available. An attachment could be added to the bot such that it would drop the seeds at a particular depth to maximise the chance of the seed germinating. This could be achieved by using a robotic arm that digs a small hole and dropping the seeds. As the bot moves forward the seeds must get covered.

Crop monitoring is essential to grow healthy crops. Now that the whole field is geo mapped each plant seed could be given a unique id. Using this unique id, each crop could be individually managed and if requirement of human supervision the plant could be easily identified.

Also watering plants at regular intervals is mandatory. The robot must be able to detect the low water content in the soil and must send data to the water pipes to turn on and water only those plants where the water has gone below the threshold value. In addition, these bots have the added advantage in accessing locations where it would be tough for humans to access.

A problem which could occur is weeds and pests. The concept of micro-spraying could significantly reduce the amount of pesticide used in crops. Micro-spraying robots use computer vision technology to detect weeds and then spray a targeted drop of pesticide onto them. To increase efficiency, the bot could be solar powered wherein a solar panel would be the roof of the bot and this would help capture maximum sunlight and also convert this into energy. Another way of removing weeds is to use high powered lasers that destroy the weeds and plants on which the laser is shown on. This can be done using computer vision technology.



**BIBLIOGRAPHY:**

1. <https://blog.robotiq.com/top-10-robotic-applications-in-the-agricultural-industry>
2. <https://components101.com/microcontrollers/stm32-nucleo-f401re-pinout-datasheet>
3. <https://www.oobautomation.com/our-services/agriculture-automation/>
4. <https://www.digikey.com/en/maker/blogs/st-nucleo-a-powerful-low-cost-alternative-to-the-arduino>
5. <https://docs.zerynth.com/latest/official/board.zerynth.st_nucleof401re/docs/index.html>
6. <https://towardsdatascience.com/what-why-and-how-of-ros-b2f5ea8be0f3?gi=276a9a277532>
7. <https://robohub.org/ros-101-intro-to-the-robot-operating-system/>