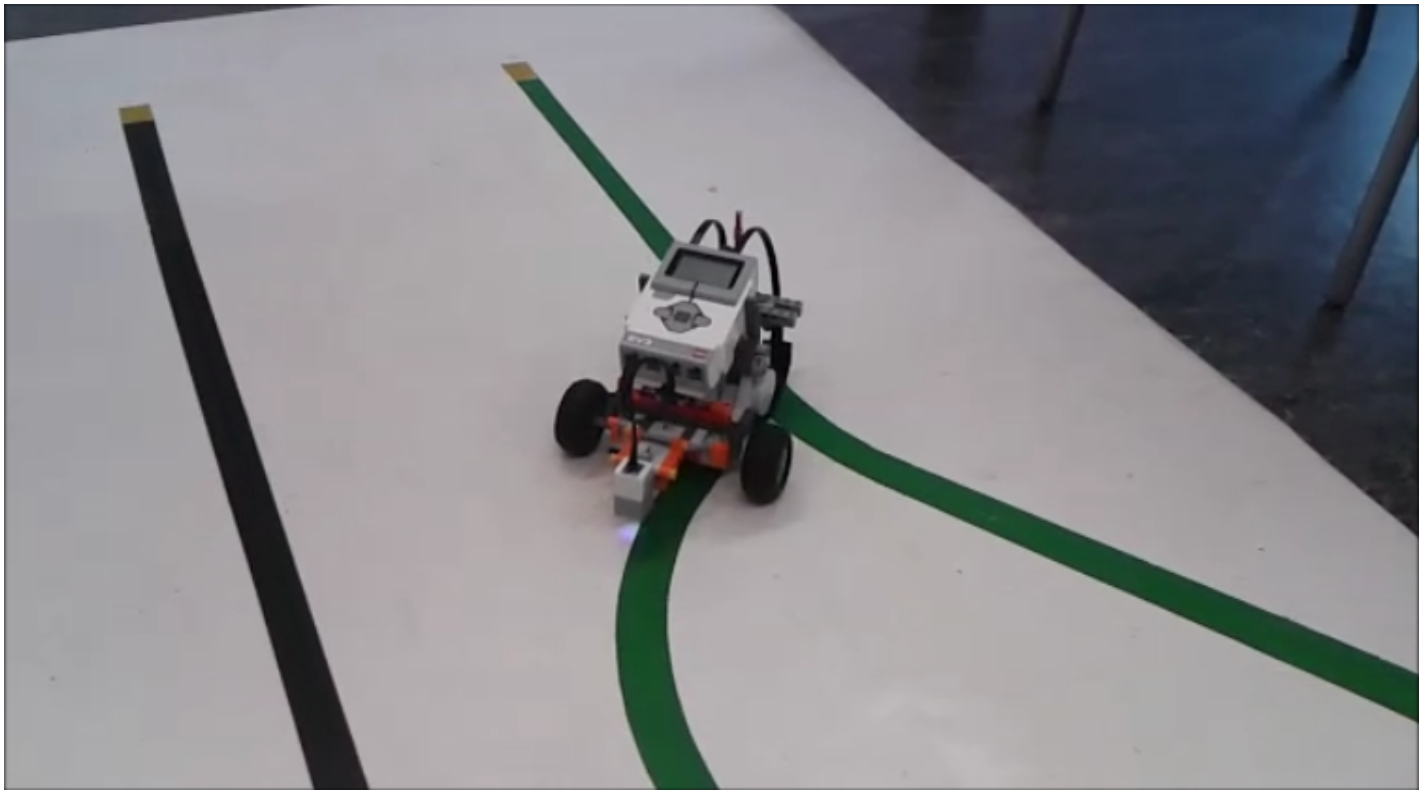
Projet Long 2015 - 2016
Robot Mindstorm



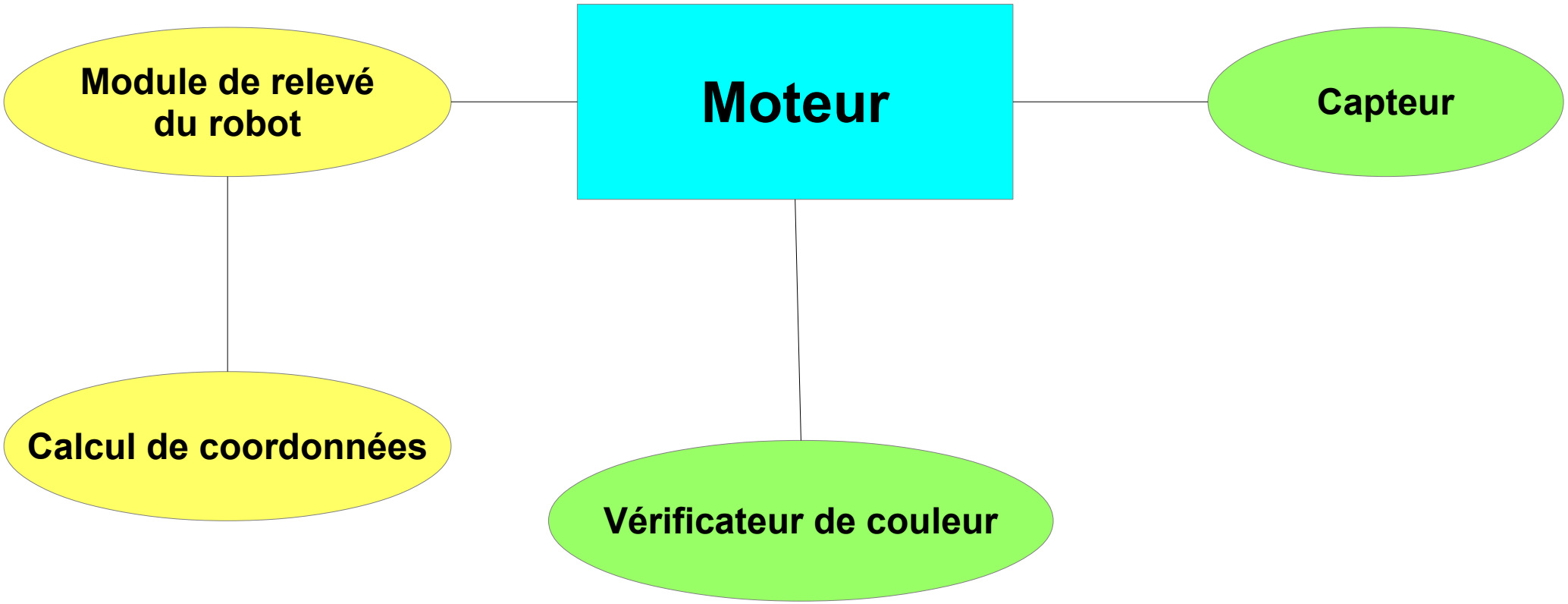
Problématique

- Problème de base :
  - Suivre un chemin prédéfini d'un point A vers un point B (le chemin n'est pas connu à l'avance)
  - Connaître son chemin
    - D'où on vient ?
    - Où va-t-on ?

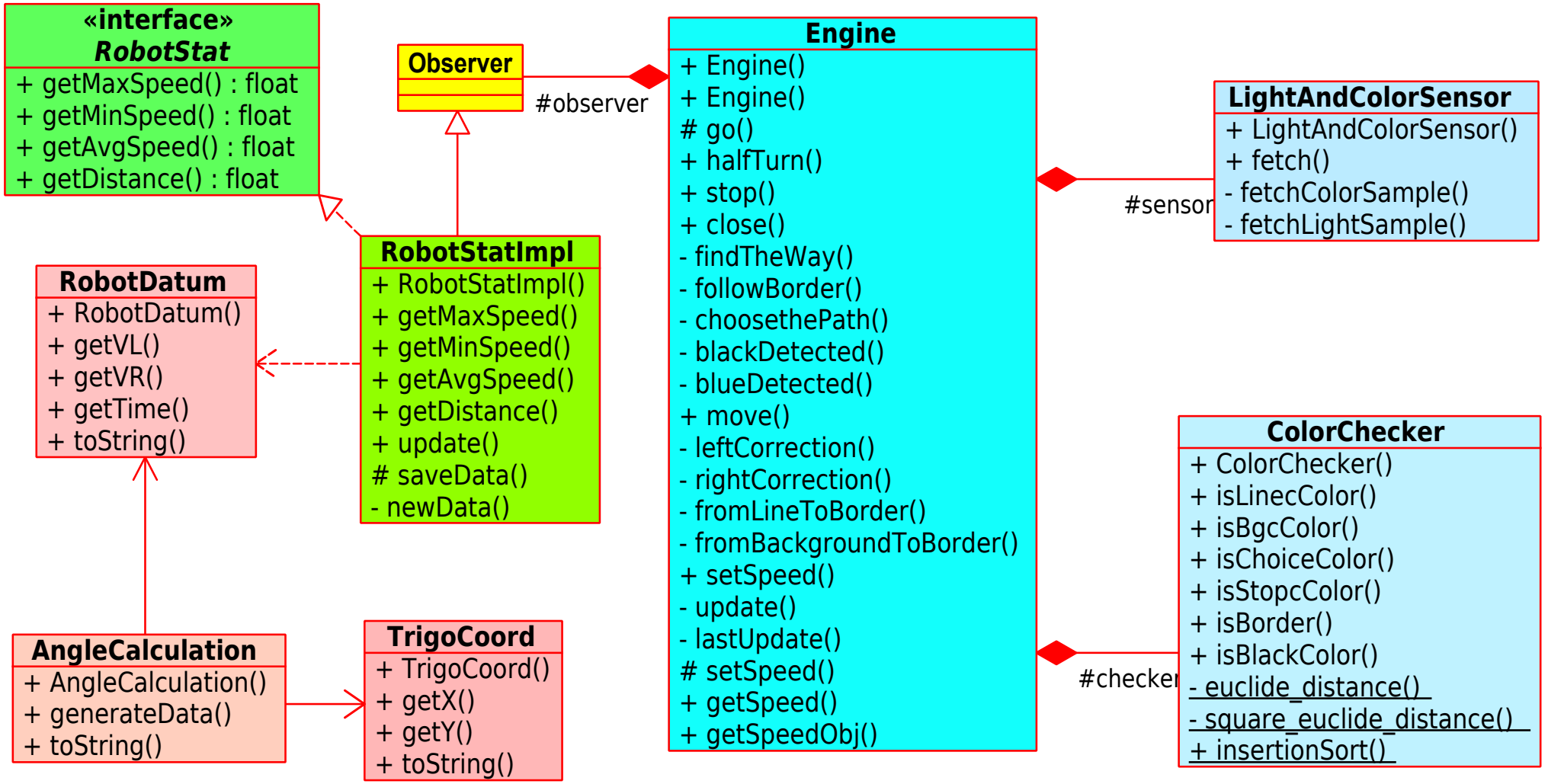
Fonctionnalités + application

- Fonctionnalités :
  - Suivi de ligne
  - Sélection de chemin
  - Génération d'un historique de navigation
- Application possible :
  - Transport de matériaux dans une usine

Architecture (1)



Architecture (2)



Architecture (3)

- Séparation entre :
  - Les actions du robot
  - La collecte d'information (capteur de couleur)
  - La capture des données courantes du robot
  - Les calculs externes

Code

```
private void newData(Engine engine) {

    long t;
    float vl, vr, velocity;
    Integer[] sp = engine.getSpeedObj();

    if (first_stat) {

        t = 0;
        current_time = System.currentTimeMillis();
        first_stat = false;

    } else {
        t = System.currentTimeMillis() - current_time;
    }

    if (Engine.half_turning)
        speed_data.add(new RobotDatum(-sp[0], sp[1], t));
    else
        speed_data.add(new RobotDatum(sp[0], sp[1], t));

    // sp[i] * 0.17165 / 360
    vl = (sp[0] * DISTANCE_2PI) / ANGLE_2PI;
    vr = (sp[1] * DISTANCE_2PI) / ANGLE_2PI;
    velocity = (vl + vr) / 2.0f;

    if (velocity > max_speed)
        max_speed = velocity;

    if (velocity < min_speed)
        min_speed = velocity;

    count++;
    avg_speed += velocity / count;
    distance += velocity * t;
}
```

```
public void generateData(ArrayList<RobotDatum> rl) {

    for (int i = 0; i < rl.size(); i++) {

        float d1 = Math.abs(rl.get(i).getVL() * DISTANCE_2PI / PI_2);
        float d2 = Math.abs(rl.get(i).getVR() * DISTANCE_2PI / PI_2);
        float[] darray = new float[] { d1, d2 };
        float[] varray = new float[] { darray[0], darray[1] };

        if (i < rl.size() - 1) {

            float difft = (rl.get(i + 1).getTime() - rl.get(i).getTime()) / 1000f;
            darray[0] = varray[0] * difft;
            darray[1] = varray[1] * difft;
        }

        float theta = (darray[0] - darray[1]) / L;
        float rad = Math.abs((darray[0] + darray[1]) / 2 * theta);

        coordinates.add(new TrigoCoord(rad, theta, rl.get(i).getTime()));
    }
}
```

Conclusion

- Ce qu'on a appris/découvert:
  - Introduction à la robotique.
  - Méthodes de développement en mode essai/erreur.
- Version 2 :
  - Améliorer la correction de trajectoire.
  - Faire un robot de course ?
- Ce qu'on aurait pu faire différemment
  - L'historique de navigation.
  - Gestion des intersections de lignes.