To convert a Twist message to individual wheel speeds for a differential drive robot, you can use the following equations:

$$\begin{aligned}
&\mathit{Linear\ Velocity} = \mathit{Twist}.\mathit{linear}.x\\
&\mathit{Angular\ Velocity} = \mathit{Twist}.\mathit{angular}.z\\
&\mathit{Wheel\ Radius} = r\\
&\mathit{Distance\ between\ Wheels} = d\\
&\mathit{LeftWheelSpeed} = \frac{(2 * \mathit{LinearVelocity} - \mathit{AngularVelocity} * d)}{2 * r}\\
&\mathit{RightWheelSpeed} = \frac{(2 * \mathit{LinearVelocity} + \mathit{AngularVelocity} * d)}{2 * r}
\end{aligned}$$

Here's how you can implement this in C++ with ROS2:

Firstly, you'll need to include the relevant headers:

```
#include "rclcpp/rclcpp.hpp"
#include "geometry_msgs/msg/twist.hpp"
```

You can then define a subscriber callback function to process the incoming Twist messages:

```
void twistCallback(const geometry_msgs::msg::Twist::SharedPtr msg)
{
  double linear_velocity = msg->linear.x;
  double angular_velocity = msg->angular.z;

  // Set the radius of your wheel and distance between the wheels
  double wheel_radius = 0.1;  // in meters
  double wheel_distance = 0.5;  // in meters

  // Calculate the left and right wheel speeds
  double left_wheel_speed = (2 * linear_velocity - angular_velocity *
wheel_distance) / (2 * wheel_radius);
  double right_wheel_speed = (2 * linear_velocity + angular_velocity *
wheel_distance) / (2 * wheel_radius);

  // Print the calculated wheel speeds
  RCLCPP_INFO(rclcpp::get_logger("TwistToWheelSpeeds"), "Left Wheel Speed: %f
m/s, Right Wheel Speed: %f m/s", left_wheel_speed, right_wheel_speed);
}
```

Finally, create a ROS2 node and set up a subscriber to get the Twist messages:

```
int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  auto node = std::make_shared<rclcpp::Node>("twist_to_wheel_speeds");

  auto subscription = node->create_subscription<geometry_msgs::msg::Twist>(
    "cmd_vel",  // Topic name
    10,         // Queue size
    twistCallback
  );

  rclcpp::spin(node);
  rclcpp::shutdown();
  return 0;
}
```

In this example, the `twistCallback` function is where the conversion happens. When a Twist message arrives on the `cmd_vel` topic, this function calculates the left and right wheel speeds based on the equations above and prints them out.

Remember to replace the `wheel_radius` and `wheel_distance` variables with the actual measurements of your specific robot.